MASTER EN SCIENCES INFORMATIQUES

Manuel utilisateur de FTTL

Mathieu, Claude; Schmitz, Alain

Award date:
1983

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX (NAMUR)

INSTITUT D'INFORMATIQUE

MANUEL UTILISATEUR DE FTTL

mémoire présenté par

Claude Mathieu

et

Alain Schmitz

en vue de l'obtention

du titre de

Licencié et maître en informatique

Année académique 1982-1983

PREFACE

Le but de ce rapport est de familiariser un utilisateur avec l'emploi du formateur "FTTL". Ce rapport a été rédigé à L'Ecole Fédérale Polytechnique de Lausanne à l'aide de l'outil qu'il décrit.

Le premier point présente une description du formateur avec ses commandes ; il reprend la logique du formatage et définit par la suite une syntaxe des commandes ; il définit également tous les termes techniques employés dans ce manuel.

Le deuxième point présente le format "standard".

Le troisième point présente les commandes du niveau global.

Le quatrième point présente les commandes du niveau local.

Le cinquième point présente l'ensemble des commandes classées par ordre alphabétique avec pour chaque commande sa syntaxe et sa sémantique.

Le sixième point présente les différents types de police et de fonte disponibles.

Le septième point présente une liste d'erreurs détectées par le formateur lors de l'introduction des commandes dans le texte à formater.

Le huitième point présente deux exemples d'utilisation des commandes du formateur ; les commandes ne sont pas toutes utilisées ; nous présentons seulement les plus courantes.

Le neuvième point présente une description du système de macros ainsi que leur méthode d'utilisation.

Pour des raisons de temps, le premier point n'a pu être rédigé à l'aide du formateur "FTTL".

TABLE DES MATIERES

INTRODUCTION A LA DESCRIPTION DU FORMATEUR

1. INTRODUCTION A LA DESCRIPTION DU FORMATEUR

1.1. Introduction

La description du formateur reprend :

- une approche à l'utilisation du formateur
- des diagrammes syntaxiques du texte au kilomètre et des commandes de formatage
- la syntaxe en notation bacchus des commandes de formatage
- un dictionnaire de données

Tous les mots entre quotes ainsi que les mots techniques utilisés dans ce point sont définis dans le dictionnaire des données.

1.2. Approche à l'utilisateur

L'utilisateur qui désire formater un document, l'introduit au terminal d'une manière continue ; c'est ce que nous appelons un "texte au kilomètre".

Un document se compose d'un ensemble d'"éléments logiques" ; nous avons défini les éléments logiques suivant :

- "titre"
- "auteur"
- "date"
- "résumé"
- "chapitre"
- "entête"
- "figure"
- "note bas de page"
- "liste"
- "paragraphe"

La syntaxe d'un texte au kilomètre est illustrée à la figure 1 (page 4).

Un certain nombre de "paramètres de présentation" du texte ont été définis. Il existe des paramètres pour l'ensemble du document (par exemple, la pagination) et des paramètres définis pour chaque élément logique (par exemple, le décalage d'une liste). Nous appelons ces paramètres de présentation des caractéristiques de formatage. L'ensemble des caractéristiques de formatage constitue ce que nous appelons le format standard (présenté au point 2).

L'utilisateur pourra au début du document se définir des caractéristiques de formatage personnelles (par exemple, demander que le document soit paginé, que le type de "fonte" pour l'élément logique de type liste soit l'italique). Ces caractéristiques de formatage s'introduisent à l'aide de commandes. Nous appelons ces commandes, commandes de niveau global. La syntaxe de ces commandes est illustrée à la figure 3 (page 5). Pour plus de détails, se référer au point 3.

Après avoir éventuellement introduit ce type de commande, l'utilisateur commencera son texte par la séquence <\DOCUMENT>.

Un texte au kilomètre se compose de caractères constituant le texte proprement dit et de commandes de formatage. Ces commandes de formatage lui permettent de spécifier les différents éléments logiques et de modifier les caractéristiques de formatage définies pour chacun de ceux-ci ; caractéristiques définies soit par le format standard, soit par les commandes du niveau global. Nous appelons ces commandes, commandes du niveau local. La syntaxe de ces commandes est illustrée à la figure 2 (page 4). Pour plus de details, se référer au point 4.

Une liste des commandes classées par ordre alphabétique et distinguant le niveau local et global est présentée au point 6.

Un exemple d'utilisation du formateur avec le texte au kilomètre et le texte imprimé/formaté est présenté au point 8.

En outre, le formateur FTTL permet l'utilisation de macros. Le mode d'emploi de ces macros ainsi qu'un exemple est présenté au point 9.

En ce qui concerne l'exécution du formatage, les commandes à utiliser sont présentées au point 8.2. Suite à l'exécution du formatage, le formateur envoie à l'écran des numéros correspondant à des erreurs dans l'introduction des commandes de formatage . Pour plus de détails, se référer au point 7.

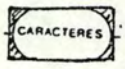
Une notation bacchus définissant le langage de formatage est présentée au point 1.4.

1.3. Analyse syntaxique d'un texte au kilomètre

La figure 1 présente la description d'un texte au kilomètre, il se compose d'un niveau global (description à la figure 3) et d'un niveau local (description à la figure 2). Le niveau local est composé d'un "niveau d'identification" (description à la figure 4) et d'un "niveau de corps" (description à la figure 5).

Les conventions graphiques sont les suivantes :

représente un nombre entier (nombre de millimètres)



représente une suite de caractères (le texte à formater proprement dit)

représente un "mot" du langage de commande\un type

représente un caractère délimiteur du langage de commande

Liste des contraintes rencontrées dans les figures :

$*_1$ la valeur (normal,gras,italique,10,...) qui suit est fonction du type de la caractéristique de formatage (fonte, centre, ...)

$*_2$ un chapitre de niveau i doit toujours suivre un chapitre de niveau inférieur ou égal à i

$*_3$ la commande de fin de note bas de page ne se rencontre que si l'élément logique courant est de type note bas de page

$*_4$ la commande de décalage ne se rencontre que si l'élément logique courant est de type paragraphe ou liste

$*_5$ la commande de modification de l'implémentation ne se rencontre que si l'élément logique courant est de type liste

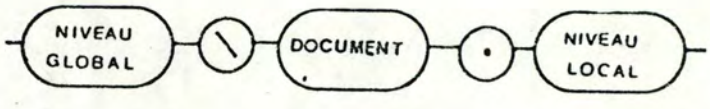


FIG. 1 - Texte au kilomètre

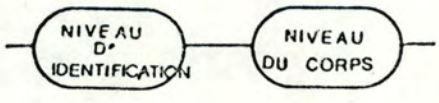


FIG. 2 - niveau local

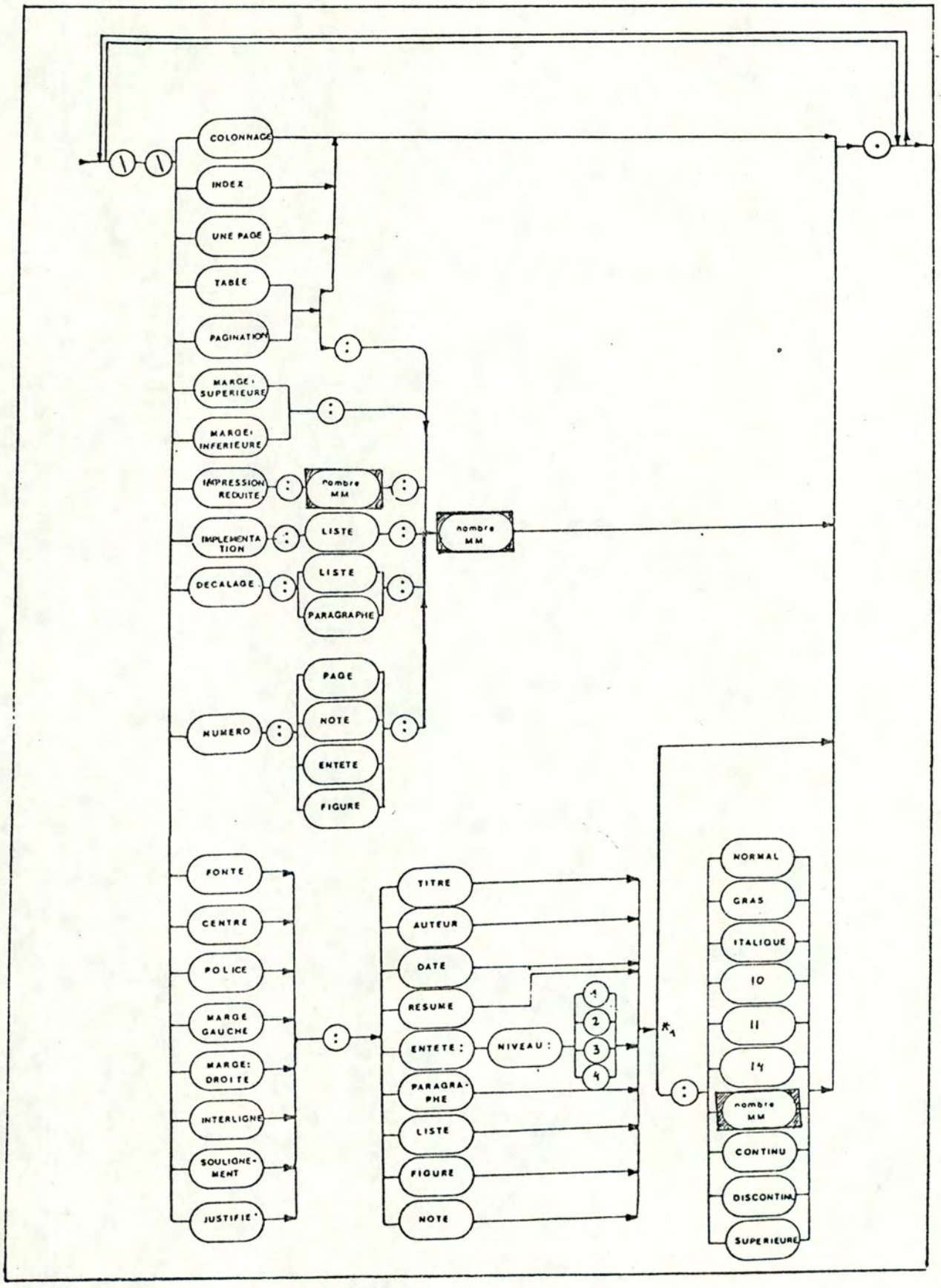


FIG. 3 - niveau global

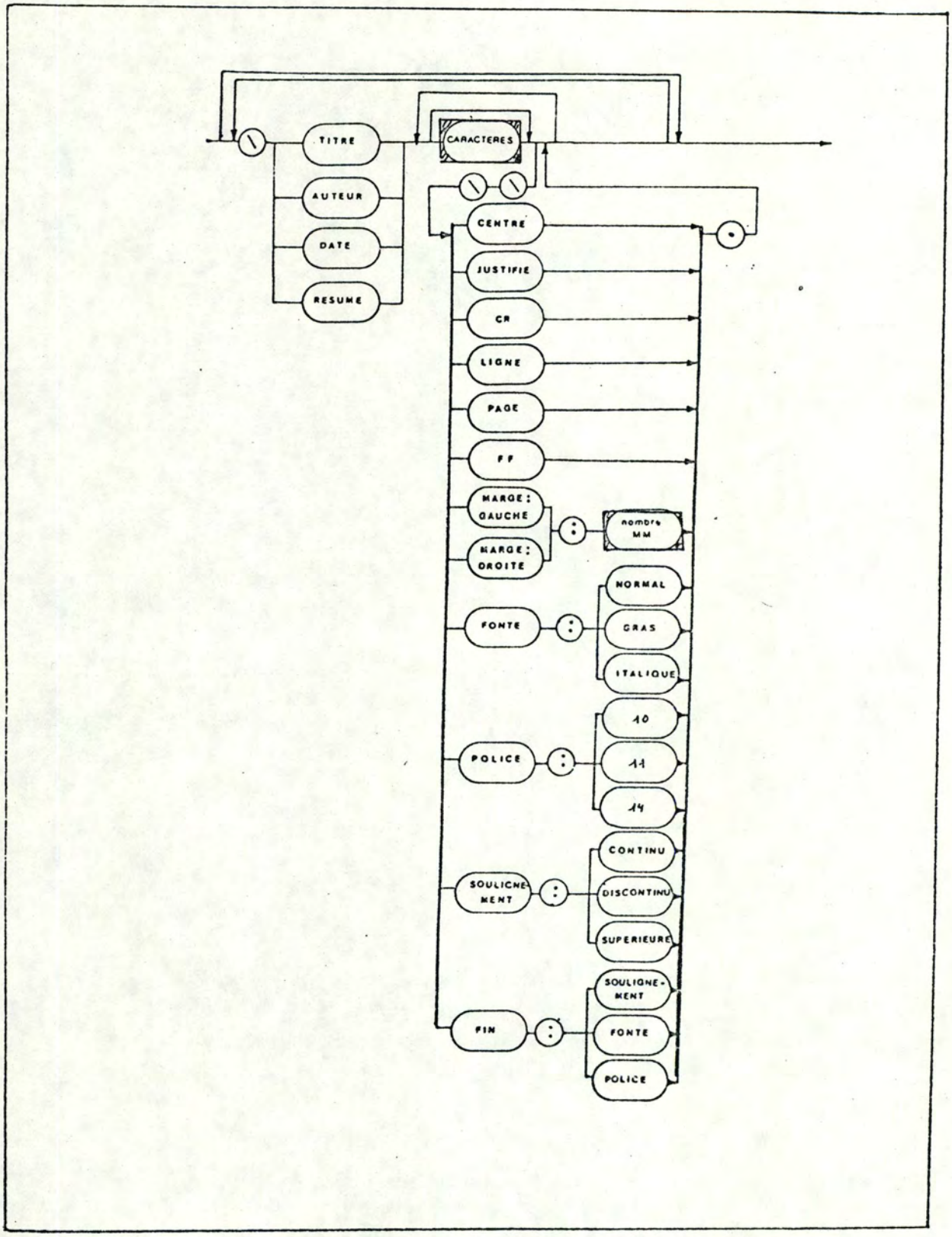


FIG. 4 - niveau d' identification

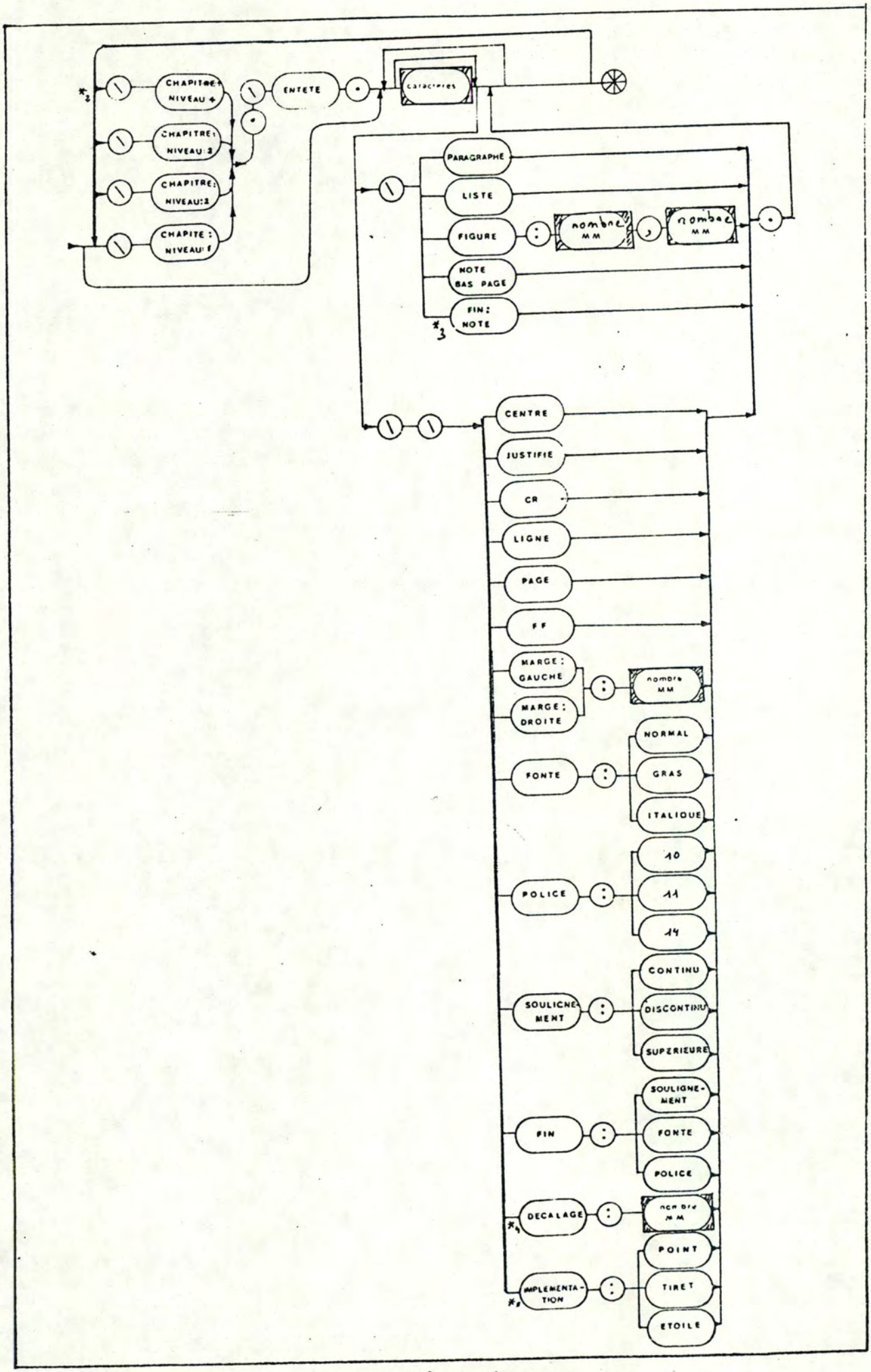


FIG. 5 - niveau du corps

1.4. Syntaxe des commandes

1.4.1. Introduction

Ce point a pour but de définir la syntaxe du langage du formateur en définissant un texte au kilomètre contenant des commandes de formatage. La syntaxe est définie dans une notation de Bacchus

1.4.2. Syntaxe

```
<TEXTE AU KILOMETRE>  ::= <COMMANDES GLOBALES> \document. <TEXTE>

<COMMANDES GLOBALES> ::= <COMMANDES GLOBALE> ! <COMMANDE GLOBALE> <COMMANDES GLOBALES>

<COMMANDE GLOBALE> ::= \\<COMMANDE>.  !
                       \\<CARACTERISTIQUE DE FORMATAGE> : <ELEMENT LOGIQUE> : <VALEUR>.  !
                       \\<CARACTERISTIQUE DE FORMATAGE>  : <ELEMENT LOGIQUE>.  !
                       \\<COMMANDE> : <VALEUR>.

<ELEMENT LOGIQUE> ::= <ELEMENT LOGIQUE D'IDENTIFICATION> ! <ELEMENT LOGIQUE DE CORPS>

<COMMANDE> ::= colonnage ! index ! pagination ! table ! une page ! impression reduite !
               marge : superieure ! marge : inferieure

<CARACTERISTIQUE DE FORMATAGE> ::= centre ! decalage ! numero ! implementation !
                                   justifie ! fonte ! police ! soulignement ! interligne !
                                   marge : gauche ! marge : droite

<TEXTE> ::= <NIVEAU D'IDENTIFICATION> <NIVEAU DU CORPS>

<NIVEAU D'IDENTIFICATION>  ::= <COMMANDES D'IDENTIFICATION LOCALE>  <NIVEAU D'IDENTIFICATION> !
                               <NIVEAU D'IDENTIFICATION>  <COMMANDES D'IDENTIFICATION LOCALES> !
                               <CARACTERES>

<COMMANDES D'IDENTIFICATION LOCALES> ::= <COMMANDE D'IDENTICATION LOCALE> !
                                         <COMMANDE D'IDENTIFICATION LOCALE> <COMMANDES D'IDENTIFICATION LOCALES>

<COMMANDE D'IDENTICATION LOCALE> ::= \<ELEMENT LOGIQUE D'IDENTIFICATION> !
                                     \\ <COMMANDE DE CARACTERISTIQUE DE FORMATAGE> !
                                     \ <COMMANDE DE FIN > ! \<COMMANDE SPECIALE>

<ELEMENT LOGIQUE D'IDENTICATION> ::= titre ! auteur ! date ! resume

<NIVEAU DU CORPS> ::= <COMMANDES DU CORPS LOCALE>  <NIVEAU DU CORPS>  !
                      <NIVEAU DU CORPS>  <COMMANDES DU CORPS LOCALES> ! <CARACTERES>

<COMMANDES DU CORPS LOCALES> ::= <COMMANDE DU CORPS LOCALE> !
                                 <COMMANDE DU CORPS LOCALE> <COMMANDES DU CORPS LOCALES>

<COMMANDE DU CORPS LOCALE> ::= \ <ELEMENT LOGIQUE DE CORPS> !
                               \\ <COMMANDE DE CARACTERISTIQUE DE FORMATAGE> !
                               \ <COMMANDE DE FIN>  ! \<COMMANDE SPECIALE>

<ELEMENT LOGIQUE DE CORPS>  ::= chapitre : niveau : <VALEUR> ! entete ! paragraphe !
                                liste ! figure ! note bas de page

<COMMANDE DE CARACTERISTIQUE DE FORMATAGE> ::= centre ! decalage ! marge : gauche !
                                               marge : droite ! justifie ! police !
                                               soulignement ! implementation ! \ ! cr !
                                               ff ! page ! fonte ! ligne

<COMMANDE DE FIN> ::= fin : soulignement ! fin : police ! fin : fonte

<COMMANDE SPECIALE> ::= `e ! `a ! `e  ! `u  ! "a  ! "e  ! "i  ! "o  ! "u  ! ^a  !^e  !  î  ! ^u

<VALEUR>  ::= superieur  ! inferieur  !  <NBMM>  ! <NBMM>,<NBMM>  !  <TYPE DE POLICE> !
              <TYPE DE FONTE> ! <TYPE DE SOULIGNEMENT>  ! <TYPE DE LISTE>

<NBMM> ::= <CHIFFRES>

<CHIFFRES> ::= <CHIFFRE>  ! <CHIFFRE>  <CHIFFRES>

<CHIFFRE> ::= 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 ! 9 ! 0

<CARACTERES> ::= <CARACTERE> ! <CARATERE> <CARACTERES>

<CARACTERE> ::= ensemble des caractères ASCII

<TYPE DE POLICE> ::= 10 ! 11 ! 14

<TYPE DE FONTE> ::= gras ! italique ! normal

<TYPE DE SOULIGNEMENT > ::= continu ! superieur ! discontinu

<TYPE DE LISTE> ::= point ! tiret ! etoile
```

1.5. Dictionnaire des données

- AUTEUR : ecrivain, homme ou femme, qui a fait un livre
- BIBLIOGRAPHIE : repertoire des ecrits reference dans le document
- CENTRAGE : texte cadre par rapport a la marge gauche et a la marge droite
- CHAPITRE : division d'un livre
- COLONNAGE : texte realise en une ou deux colonnes par page
- DATE : indication du jour du mois et de l'annee
- DECALAGE : nombre de millimetres de decalage de la premiere ligne d'un paragraphe ou des elements d'une liste
- ELEMENT LOGIQUE : element de structure d'un livre, ces elements etant : auteur, titre, date, resume, chapitre, entete, liste, figure, note bas de page, paragraphe
- ENTETE : titre de chapitre
- FIGURE : dessin servant a la representation
- FONTE : style de caracteres : italique, normal, gras
- IMPLEMENTATION D'UNE LISTE : type d'element desire comme premier caractere d'un element d'une liste : point, tiret, etoile, ...
- IMPRESSION REDUITE : indication que l'on ne desire qu'un nombre limite de feuille(s) imprimee(s)
- INDEX : table alphabetique de noms cites dans le document accompagnes de references
- INTERENTITE : nombre de millimetres a laisser entre deux entites
- INTERLIGNE : nombre de millimetres a laisser entre deux lignes
- JUSTIFIE : lignes traitees de facon a avoir un alignement a gauche et a droite
- LARGEUR DE FEUILLE : nombre de millimetres de la feuille en largeur
- LISTE : suite de noms
- LONGUEUR DE FEUILLE : nombre de millimetres de la feuille en longueur
- MARGE DROITE : nombre de millimetres laisses a la droite du texte
- MARGE INFERIEURE : nombre de millimetres laisses vides en bas de page
- MARGE GAUCHE : nombre de millimetres laisses vides a la gauche du texte
- MARGE SUPERIEURE : nombre de millimetres laisses vides en haut de la page
- PAGINATION : mise d'un numero croissant sur chacune des pages du document
- PARAGRAPHE : petite section d'un chapitre
- PARAMETRE DE PRESENTATION : caracteristique typographique, les differentes caracteristiques typographiques etant : le centrage, le colonnage, le decalage, la fonte, l'implementation d'une liste, l'impression reduite, l'index, l'interligne, l'interentite, la justification, la largeur et la longueur d'une feuille, la marge gauche, la marge droite, la marge inferieure, la marge superieure, la pagination, la police, le saut de page et de ligne, le soulignement, la table des matieres et le type de feulle
- POLICE : grandeur des caracteres
- RESUME : abrege, sommaire
- SAUT DE PAGE : indication de passage a la page suivante
- SAUT DE LIGNE : indication de passage a la ligne suivante
- SOULIGNEMENT : methode de mise en evidence, differents types de soulignement : continu, discontinu, superieur
- TABLE DES MATIERES : enumeration des chapitres, des questions traitees dans le document
- TEXTE AU KILOMETRE : suite de caracteres sans retour de chariot
- TYPE DE FEUILLE : genre de feuille pour laquelle on desire un formatage, le type de feuille implique une hauteur et une largeur

DESCRIPTION DU FORMAT "STANDARD"

2. DESCRIPTION DU FORMAT "STANDARD"

2.1. Introduction

Pour chaque élément logique sont définies des indications typographiques de formatage qui caractérisent la présentation des différents éléments logiques sur la feuille imprimée.

Ce chapitre comprend :

- une liste d'indications de formatage pour les différents éléments logiques
- une liste des indications de formatage par élément logique

2.2. Liste des indications de formatage pour les éléments logiques

le saut de page : un élément logique saute de page s' il est imprimé sur la feuille suivante

le centrage : un élément logique est centré si l'espace entre le bord gauche de la feuille imprimée du document et le premier caractère à gauche de l'élément logique est identique à l'espace entre le bord droit de la feuille imprimée du document et le dernier caractère à droite de l'élément logique

l'interligne : nombre de millimètres entre les lignes d'un élément logique

la marge gauche : nombre de millimètre(s) entre l'élément logique et le bord gauche de la feuille imprimée

la marge droite : nombre de millimètre(s) entre l'élément logique et le bord droite de la feuille imprimée

la marge supérieure : nombre de millimètre(s) entre l'élément logique et le bord supérieure de la feuille imprimée

la marge inférieure : nombre de millimètre(s) entre l'élément logique et le bord inférieure de la feuille imprimée

la fonte des caractères : type caractérisant la fonte des caractères d'un élément logique (nb. 1)

la police des caractères : entier caractérisant la police des caractères d'un élément logique (nb. 2)

le soulignement : entier caractérisant le type de soulignement d'un élément logique (les différents types de soulignement sont le continu, le discontinu et le supérieur.)

NB : 1 les différents types de fonte sont donnés au point 6

NB : 2 les différents types de police sont donnés au point 6

2.3. Liste des indications de formatage par élément logique

Le titre

l'élément logique titre ne saute pas de page
l'élément logique titre est centré sur la page
l'interligne de l'élément logique titre est fixé à 5 millimètres
la marge gauche de l'élément logique titre est fixée à 30 millimètres
la marge droite de l'élément logique titre est fixée à 10 millimètres
le type de fonte des caractères de l'élément logique titre est le gras
la police des caractères de l'élément logique titre est celle correspondant au numéro 14
l'élément logique titre n'est pas souligné

L'auteur

l'élément logique auteur ne saute pas de page
l'élément logique auteur est centré sur la page
l'interligne de l'élément logique auteur est fixé à 3 millimètres
la marge gauche de l'élément logique auteur est fixée à 30 millimètres
la marge droite de l'élément logique auteur est fixée à 10 millimètres
le type de fonte des caractères de l'élément logique auteur est l'italique
la police des caractères de l'élément logique auteur est celle correspondant au numéro 11
l'élément logique auteur n'est pas souligné

La date

l'élément logique date ne saute pas de page
l'élément logique date est centré sur la page
l'interligne de l'élément logique date est fixé à 4 millimètres
la marge gauche de l'élément logique date est fixée à 30 millimètres
la marge droite de l'élément logique date est fixée à 10 millimètres
le type de fonte des caractères de l'élément logique date est l'italique
la police des caractères de l'élément logique date est celle correspondant au numéro 11
l'élément logique date n'est pas souligné

Le résumé

l'élément logique résumé ne saute pas de page
l'élément logique résumé est centré sur la page
l'interligne de l'élément logique résumé est fixé à 2 millimètres
la marge gauche de l'élément logique résumé est fixée à 10 millimètres
la marge droite de l'élément logique résumé est fixée à 0 millimètre
le type de fonte des caractères de l'élément logique résumé est le normal
la police des caractères de l'élément logique résumé est celle correspondant au numéro 11
l'élément logique résumé n'est pas souligné

L'entête de niveau 1

l'élément logique entête de niveau 1 ne saute pas de page
l'élément logique entête de niveau 1 est centré sur la page
l'interligne de l'entête de niveau 1 est fixé à 3 millimètres
la marge gauche de l'élément logique entête de niveau 1 est fixée à 10 millimètres
la marge droite de l'élément logique entête de niveau 1 est fixée à 0 millimètre
le type de fonte des caractères de l'élément logique entête de niveau 1 est le gras
la police des caractères de l'élément logique entête de niveau 1 est celle correspondant au numéro 14
l'élément logique entête de niveau 1 est souligné en continu

L'entête de niveau supérieure à 1

l'élément logique entête de niveau supérieur à 1 ne saute pas de page
l' élément logique entête de niveau supérieur à 1 n'est pas centré sur la page
l'interligne de l'entête de niveau supérieur à 1 est fixé à 3 millimètres
la marge gauche de l'élément logique entête de niveau supérieur à 1 est fixée à 10 millimètres
la marge droite de l'élément logique entête de niveau supérieur à 1 est fixée à 0 millimètre
le type de fonte des caractères de l'élément logique entête de niveau supérieur à 1 est le normal
la police des caractères de l'élément logique entête de niveau supérieur à 1 est celle correspondant au numéro 11
l'élément logique entête de niveau supérieur à 1 est souligné en continu

Le paragraphe

l'élément logique paragraphe ne saute pas de page
l'élément logique paragraphe n'est pas centré sur la page
l'interligne de l'élément logique paragraphe est fixé à 1 millimètre
la marge gauche de l'élément logique paragraphe est fixée à 0 millimètre
la marge droite de l'élément logique paragraphe est fixée à 10 millimètres
le type de fonte des caractères de l'élément logique paragraphe est le normal
la police des caractères de l'élément logique paragraphe est celle correspondant au numéro 11
l'élément logique paragraphe n'est pas souligné
la première ligne de l'élément logique paragraphe est décalée vers la droite de 10 millimètres

La liste

l'élément logique liste ne saute pas de page
l'élément logique liste n'est pas centré sur la page
l'interligne de l'élément logique liste est fixé à 3 millimètres
la marge gauche de l'élément logique liste est fixée à 10 millimètres
la marge droite de l'élément logique liste est fixée à 0 millimètre
le type de fonte des caractères de l'élément logique liste est le normal
la police des caractères de l'élément logique liste est celle correspondant au numéro 11
l'élément logique liste n'est pas souligné
chaque composant de l'élément logique liste est décalé par rapport à la marge gauche de 10 millimètres
chaque composant de l'élément logique liste est précédé d'un tiret

La note bas de page

l'élément logique note bas de page ne saute pas de page
l'élément logique note bas de page n'est pas centré sur la page
l'interligne de l'élément logique note bas de page est fixé à 1 millimètre
la marge gauche de l'élément logique note bas de page est fixée à 10 millimètres
la marge droite de l'élément logique note bas de page est fixée à 0 millimètre
le type de fonte des caractères de l'élément logique note bas de page est le normal
la police des caractères de l'élément logique note bas de page est celle correspondant au numéro 10
l'élément logique note bas de page n'est pas souligné
la première occurrence de l'élément logique note bas de page, de même que sa référence a le numéro 1

La figure

l'élément logique figure ne saute pas de page
l'élément logique figure est centré sur la page
l'interligne de l'élément logique figure est fixé à 1 millimètre
la marge gauche de l'élément logique figure est fixée à 10 millimètres
la marge droite de l'élément logique figure est fixée à 0 millimètre
le type de fonte des caractères de l'élément logique figure est l'italique
la police des caractères de l'élément logique figure est celle correspondant au numéro 10
l'élément logique 'figure n'est pas souligné
la première occurrence de l'élément logique figure, de même que sa référence a le numéro 1

Outre la définition des indications de formatage pour les éléments logiques, certains paramètres du document ainsi que des "utilitaires" sont initialisés :

le document est imprimé en simple colonne
le document n'est pas paginé
toutes les pages composant le document sont imprimées
le document ne comporte ni table des matières, ni index
les éléments d'identifications ne se trouvent pas centrés sur la première page du document
la longueur d'une feuille du document est fixée à 296 millimètres
la largeur d'une feuille du document est fixée à 215 millimètres
l'écart entre les éléments logiques est fixé à 5 millimètres
la marge supérieure du document est fixée à 20 millimètres
la marge inférieure du document est fixée à 20 millimètres
le format d'une feuille du document est de type A4

Dans le cas où une table des matières serait demandée, elle serait définie par défaut de la manière suivante :

l'interligne serait de 3 millimètres
la marge gauche serait de 20 millimètres
la marge droite serait de 10 millimètres
la fonte des caractères serait la fonte normale
la police des caractères serait la police numéro 11
aucun élément de la table ne serait souligné
les entêtes de niveau 1 et de niveau 2 seraient uniquement retenues

DESCRIPTION DES COMMANDES DU NIVEAU GLOBAL

3. Description et utilisation des commandes du niveau global

3.1. Introduction

L'utilisateur qui désire modifier une ou plusieurs indication(s) de formatage pour un ou plusieurs élément(s) logique(s) introduira une ou plusieur(s) commande(s) avant le début du texte à formater.

Les commandes se présentent sous trois formes différentes :

- le premier type de commande permet au formateur de créer des "utilitaires" en plus du formatage des éléments logiques
- le deuxième type de commande permet d'assigner de nouvelles valeurs aux caractéristiques de formatage définies par élément logique
- le troisième type de commande permet d'assigner de nouvelles valeurs aux caractéristiques de formatage définies pour le document

Les trois types de commandes sont explicités de la manière suivante : pour chaque commande :

- description du format général de la commande
- fonction de la commande
- la liste complète des différentes commandes avec leur séquence ainsi que leur effet

3.2. Description du premier type de commande

1. format de la commande :

\\<COMMANDE>

2. fonction :

Ce type de commande permet au formateur de créer des "utilitaires"

3. liste des utilitaires :

Séquence : **\\CO(lonnage).**

Effet : le texte sera formaté en double colonne

Séquence : **\\IN(dex).** **(*)**

Effet : cette commande permet d'obtenir un index qui reprendra tous les mots annotés par l'utilisateur ainsi que leur(s) numéro(s) de page(s) correspondant(s); le document sera paginé, le numéro se trouvant en bas de page

Séquence : **\\PAG(gination).**

Effet : cette commande permet de paginer le document ; le numéro se trouvera en bas de page, sera centré et aura la forme : "- NUMERO -"

Séquence : **\\TA(ble des matières).**

Effet : cette commande permet d'obtenir une table des matières qui comprendra la liste des entêtes de chapitres ainsi que leur numéro de page correspondant ; le document sera paginé ; le numéro se trouvera en bas de page, sera centré et aura la forme : "- NUMERO -"

Séquence : **\\UN(ne page).**

Effet : les éléments logiques d'identification (titre(s), auteur(s), date(s), résumé) se trouveront centrés sur la première page du document ; cette page ne sera pas numérotée même si la pagination est demandée. Les éléments d'identification ne peuvent remplir plus d'une page

3.3. Description du deuxième type de commande

1. Format de la commande :

\\<CARACTERISTIQUE DE FORMATAGE> : <ELEMENT LOGIQUE> : <VALEUR>.

ou

\\<CARACTERISTIQUE DE FORMATAGE> : <ELEMENT LOGIQUE>.

2. Fonction :

Cette commande modifie les indications de formatage pré-définies en leur assignant une nouvelle VALEUR et ce pour un élément logique défini

3. Liste des commandes

Séquences :

\\CE(ntre) : AU(teur).
\\CE(ntre) : DA(te).
\\CE(ntre) : RE(sumé).
\\CE(ntre) : TI(tre).
\\CE(ntre) : EN(tête) : (niveau) : 1.
\\CE(ntre) : EN(tête) : (niveau) : 2.
\\CE(ntre) : EN(tête) : (niveau) : 3.
\\CE(ntre) : EN(tête) : (niveau) : 4.
\\CE(ntre) : LI(ste).
\\CE(ntre) : NO(te bas de page).
\\CE(ntre) : PA(ragraphe).
\\CE(ntre) : TA(ble des matières).
\\CE(ntre) : FI(gure).

Effet : les éléments logiques spécifiés seront centrés

Séquence : **\\DE(calage) : LI(ste) : VALEUR.**

Effet : cette commande permet de décaler toute les lignes de(s) liste(s) d'un certain nombre de millimètres vers la droite ; VALEUR étant ce nombre

Séquence : **\\DE(calage) : PA(ragraphe) : VALEUR.**

Effet : cette commande permet de décaler la première ligne de(s) paragraphe(s) d'un certain nombre de millimètres ; VALEUR est ce nombre

Séquence : **\\NU(mero) : N(ote) : VALEUR.**

Effet : cette commande permet d'initialiser le numéro de(s) note(s) bas de page ainsi que leur référence ; VALEUR est le numéro de la première note et de sa première référence

Séquence : \\IMP(plémentation) : L(iste) : TYPE.

Effet : cette commande permet de pré-fixer tous les éléments de(s) liste(s) par un caractère TYPE ; les différents types disponibles sont le point, le tiret ou l'étoile ; TYPE prend les valeurs :

- **T(iret)**
- **P(oint)**
- **E(toile)**

Séquence : \\NU(mero) : F(igure) : VALEUR.

Effet : cette commande permet d'initialiser le numéro de(s) figure(s) ainsi que leur référence ; VALEUR est le numéro de la première figure et de sa première référence

Séquence : \\NU(mero) : P(age) : VALEUR.

Effet : cette commande permet d'initialiser le numéro de page ; VALEUR est le numéro de la première page

Séquence : \\NU(mero) : E(ntête : niveau 1) : VALEUR.

Effet : cette commande permet d'initialiser le numéro de(s) entête(s) de niveau 1 ; VALEUR est le numéro de la première entête de niveau 1

Séquences :

\\JU(stifie) : AU(teur).
\\JU(stifie) : DA(te).
\\JU(stifie) : RE(sumé).
\\JU(stifie) : TI(tre).
\\JU(stifie) : EN(tête) : (niveau) : 1.
\\JU(stifie) : EN(tête) : (niveau) : 2.
\\JU(stifie) : EN(tête) : (niveau) : 3.
\\JU(stifie) : EN(tête) : (niveau) : 4.
\\JU(stifie) : LI(ste).
\\JU(stifie) : NO(te bas de page).
\\JU(stifie) : PA(ragraphe).
\\JU(stifie) : TA(ble des matières).
\\JU(stifie) : FI(gure).

Effet : ces commandes permettent de justifier à gauche et à droite tous les éléments logiques spécifiés

Séquences :

\\FO(nte) : AU(teur) : TYPE.
\\FO(nte) : DA(te) : TYPE.
\\FO(nte) : RE(sumé) : TYPE.
\\FO(nte) : TI(tre) : TYPE.
\\FO(nte) : EN(tête) : (niveau) : 1 : TYPE.
\\FO(nte) : EN(tête) : (niveau) : 2 : TYPE.
\\FO(nte) : EN(tête) : (niveau) : 3 : TYPE.
\\FO(nte) : EN(tête) : (niveau) : 4 : TYPE.
\\FO(nte) : LI(ste) : TYPE.
\\FO(nte) : NO(te) bas de page) : TYPE.
\\FO(nte) : PA(ragraphe) : TYPE.
\\FO(nte) : TA(ble des matières) : TYPE.
\\FO(nte) : FI(gure) : TYPE.

Effet : ces commandes permettent de définir le type de fonte pour les éléments logiques spécifiés (nb. 1)

Séquences :

\\PO(lice) : AU(teur) : TYPE.
\\PO(lice) : DA(te) : TYPE.
\\PO(lice) : RE(sumé) : TYPE.
\\PO(lice) : TI(tre) : TYPE.
\\PO(lice) : EN(tête) : (niveau) : 1 : TYPE.
\\PO(lice) : EN(tête) : (niveau) : 2 : TYPE.
\\PO(lice) : EN(tête) : (niveau) : 3 : TYPE.
\\PO(lice) : EN(tête) : (niveau) : 4 : TYPE.
\\PO(lice) : LI(ste) : TYPE.
\\PO(lice) : NO(te) bas de page) : TYPE.
\\PO(lice) : PA(ragraphe) : TYPE.
\\PO(lice) : TA(ble des matières) : TYPE.
\\PO(lice) : FI(gure) : TYPE.

Effet : ces commandes permettent de définir le type de police pour les éléments logiques spécifiés (nb. 2)

Séquences :

\\SO(ulignement) : AU(teur) : TYPE.
\\SO(ulignement) : DA(te) : TYPE.
\\SO(ulignement) : RE(sumé) : TYPE.
\\SO(ulignement) : TI(tre) : TYPE.
\\SO(ulignement) : EN(tête) : (niveau) : 1 : TYPE.
\\SO(ulignement) : EN(tête) : (niveau) : 2 : TYPE.
\\SO(ulignement) : EN(tête) : (niveau) : 3 : TYPE.
\\SO(ulignement) : EN(tête) : (niveau) : 4 : TYPE.
\\SO(ulignement) : LI(ste) : TYPE.
\\SO(ulignement) : NO(te bas de page) : TYPE.
\\SO(ulignement) : PA(ragraphe) : TYPE.
\\SO(ulignement) : TA(ble des matières) : TYPE.
\\SO(ulignement) : FI(gure) : TYPE.

NB: 1 les différents TYPES de fonte sont donnés au point 6

NB: 2 les différents TYPES de police sont donnés au point 6

Effet : ces commandes permettent de définir le type de soulignement pour les éléments logiques spécifiés ; les différents types de soulignement sont le continu, le discontinu ou le supérieur ; TYPE peut prendre les valeurs :

- **C(ontinu)**
- **D(iscontinu)**
- **S(upérieur)**

Séquences :

```
\\INT(erligne) : AU(teur) : VALEUR.
\\INT(erligne) : DA(te) : VALEUR.
\\INT(erligne) : RE(sumé) : VALEUR.
\\INT(erligne) : TI(tre) : VALEUR.
\\INT(erligne) : EN(tête) : (niveau) : 1 : VALEUR.
\\INT(erligne) : EN(tête) : (niveau) : 2 : VALEUR.
\\INT(erligne) : EN(tête) : (niveau) : 3 : VALEUR.
\\INT(erligne) : EN(tête) : (niveau) : 4 : VALEUR.
\\INT(erligne) : LI(ste) : VALEUR.
\\INT(erligne) : NO(te) bas de page) : VALEUR.
\\INT(erligne) : PA(ragraphe) : VALEUR.
\\INT(erligne) : TA(ble des matières) : VALEUR.
\\INT(erligne) : FI(gure) : VALEUR.
```

Effet : ces commandes permettent de définir un certain nombre de millimètres entre les lignes des éléments logiques spécifiés ; VALEUR est ce nombre

Séquences :

```
\\MA(rge) : G(auche) : AU(teur) : VALEUR.
\\MA(rge) : G(auche) : DA(te) : VALEUR.
\\MA(rge) : G(auche) : RE(sumé) : VALEUR.
\\MA(rge) : G(auche) : TI(tre) : VALEUR.
\\MA(rge) : G(auche) : EN(tête) : (niveau) : 1 : VALEUR.
\\MA(rge) : G(auche) : EN(tête) : (niveau) : 2 : VALEUR.
\\MA(rge) : G(auche) : EN(tête) : (niveau) : 3 : VALEUR.
\\MA(rge) : G(auche) : EN(tête) : (niveau) : 4 : VALEUR.
\\MA(rge) : G(auche) : LI(ste) : VALEUR.
\\MA(rge) : G(auche) : (NOte bas de page) : VALEUR.
\\MA(rge) : G(auche) : PA(ragraphe) : VALEUR.
\\MA(rge) : G(auche) : TA(ble des matières) : VALEUR.
\\MA(rge) : G(auche) : FI(gure) : VALEUR.
```

Effet : ces commandes permettent de définir un certain nombre de millimètres pour la marge gauche des éléments logiques spécifiés ; VALEUR est ce nombre

Séquences :

\\MA(rge) : D(roite) : AU(teur) : VALEUR.
\\MA(rge) : D(roite) : DA(te) : VALEUR.
\\MA(rge) : D(roite) : RE(sumé) : VALEUR.
\\MA(rge) : D(roite) : TI(tre) : VALEUR.
\\MA(rge) : D(roite) : EN(tête) : (niveau) : 1 : VALEUR.
\\MA(rge) : D(roite) : EN(tête) : (niveau) : 2 : VALEUR.
\\MA(rge) : D(roite) : EN(tête) : (niveau) : 3 : VALEUR.
\\MA(rge) : D(roite) : EN(tête) : (niveau) : 4 : VALEUR.
\\MA(rge) : D(roite) : LI(ste) : VALEUR.
\\MA(rge) : D(roite) : NO(te bas de page) : VALEUR.
\\MA(rge) : D(roite) : PA(ragraphe) : VALEUR.
\\MA(rge) : D(roite) : TA(ble des matières) : VALEUR.
\\MA(rge) : D(roite) : FI(gure) : VALEUR.

Effet : ces commandes permettent de définir un certain nombre de millimètres pour la marge droite des éléments logiques spécifiés ; VALEUR est ce nombre

3.4. Description du troisième type de commande

1. Format de la commande :

\\<COMMANDE> : <VALEUR>.

2. Fonction :

Ce type de commande modifie les indications de formatage pré-définies en leur assignant une nouvelle VALEUR et ce pour tout le document

3. Liste des commandes :

Séquence : **\\IMP(ression réduite) : NUMERO 1 , NUMERO 2.**

Effet : cette commande permet de spécifier qu'un nombre réduit de pages doit être imprimé.

Séquence : **\\EC(art entre éléments logiques) : VALEUR.**

Effet : cette commande permet de définir un nombre de millimètres d'espacement entre les éléments logiques ; VALEUR est ce nombre

Séquences :

\\MA(rge) : S(upérieure) : VALEUR.
\\MA(rge) : I(nférieure) : VALEUR.

Effet : ces commandes permettent de définir un nombre de millimètres pour la marge inférieure et supérieure des pages du document ; VALEUR est ce nombre

Séquences :

\\PA(gination) : S(upérieure).
\\PA(gination) : I(nférieure).

Effet : ces commandes permettent de définir une numérotation en haut ou en bas de page ; le numéro sera centré et aura la forme : "- NUMERO -"

Séquence : **\\TA(ble des matières) : VALEUR.**

Effet : cette commande permet de spécifier le nombre de niveaux d'entête de chapitre qui seront repris dans la table des matières ; VALEUR est ce nombre ; il doit être compris entre 1 et 4. Une table des matières sera créée

Séquence : **\\TY(pe feuille) : VALEUR.**

Effet : cette commande permet de spécifier le type de feuille qui sera imprimée ; VALEUR ne peut prendre que le type A4 ; le type A4 détermine une feuille de 296 millimètres de hauteur et de 210 millimètres de largeur

CREATION DES ELEMENT LOGIQUES

DESCRIPTION ET UTILISATION DES COMMANDES DU NIVEAU LOCAL

4. CREATION DES ELEMENTS LOGIQUES - DESCRIPTION ET UTILISATION DES COMMANDES DU NIVEAU LOCAL

4.1. Introduction

Le texte à formater commence obligatoirement par la séquence :

\DO(cument).

Le texte source comprendra un certain nombre de commandes pour le formatage ; celles-ci seront de deux types :

- le premier type de commande permet de spécifier la structure du document
- le deuxième type de commande permet de modifier les caractéristiques de formatage définies soit par les commandes du niveau global, soit par le format "standard". Ces commandes portent sur l'élément logique courant et le concernent soit dans son entierté soit en partie

4.2. Description des commandes concernant la structure du document

4.2.1. Introduction

Un document se compose de deux parties, chacune étant constituée d'éléments logiques. La première de ces parties reprend l'identification du document ; ses éléments logiques sont :

- le(s) titre(s)
- nom(s) de(s) auteur(s)
- le(s) date(s)
- le résumé

La deuxième partie constitue le corps du document propement dit. Le corps du document sera une suite de chapitres de niveau 1 ; chacun de ceux-ci se composera d'une entête suivie d'un corps de chapitre. Le corps de chapitre sera

* soit une suite d'éléments logiques, c'est-à-dire :

  - paragraphe
  - liste
  - note bas de page
  - figure

* soit un chapitre de niveau inférieur ayant la même structure que le chapitre de niveau 1

4.2.2. Commandes définissant les éléments d'identification

1. Format de la commande :

\<COMMANDE>.

2. Fonction :

Ce type de commande permet de créer un élément logique constituant l'identification du document

3. Liste des commandes :

Séquence : **\TI(tre).**

Effet : cette commande détermine que les caractères suivant la séquence jusqu'à la séquence suivante constituent le titre du document

Séquence : **\AU(teur).**

Effet : cette commande détermine que les caractères suivant la séquence jusqu'à la séquence suivante constituent le nom de l'auteur

Séquence : **\DA(te).**

Effet : cette commande détermine que les caractères suivant la séquence jusqu'à la séquence suivante constituent la date

Séquence : **\RE(sumé)**

Effet : cette commande détermine que les caractères suivant la séquence jusqu'à la séquence suivante constituent le résumé

4.2.3. Commande définissant le corps du document

1. Format de la commande :

\<COMMANDE>.

2. Fonction :

Ce type de commande permet de créer un élément logique constituant le corps du document

3. Liste des commandes :

Séquence : **\CH(apitre) (: niveau) : VALEUR.**

Effet : cette commande permet de créer un chapitre d'un niveau déterminé ; VALEUR est ce niveau et est compris entre 1 et 4

Séquence : **\EN(tête).**

Effet : cette commande détermine que les caractères qui suivent jusqu'à la séquence suivante ou la fin du texte, constituent une entête. La numérotation est faite automatiquement en fonction du niveau du chapitre

Séquence : **\FI(gure) : X , Y : 'NOM'** .(*)

Effet : cette commande détermine que les caractères qui suivent jusqu'à la séquence suivante ou la fin du texte, constituent le texte d'une figure de largeur X et de longueur Y , X et Y sont exprimés en nombre de millimètres ; NOM correspond au fichier contenant la figure. L'insertion de la figure n'étant pas réalisée, la commande reserve uniquement la place ; le texte se situera en-dessous de la figure ; celle-ci sera numérotée et aura une référence dans le texte. La commande utilisée actuellement a la forme * :

\FI(gure) : X , Y.

Séquence : **\LI(ste).**

Effet : cette commande détermine que les caractères qui suivent jusqu'à la séquence suivante ou la fin du texte, constituent une liste. Chaque élément de cette liste est séparé par trois boas ; un élément de la liste est toujours précédé par un caractère correspondant à un type d'implémentation défini

Séquence : **\NO(te bas de page).**

Effet : cette commande détermine que les caractères qui suivent jusqu'à la séquence suivante ou la fin du texte ou la séquence**\F(in) : NO(te),** constituent une note bas de page. La note sera placée en bas de page avec sa référence dans le texte

Séquence : **\PA(ragraphe).**

Effet : cette commande détermine que les caractères qui suivent jusqu'à la séquence suivante ou la fin du texte, constituent un paragraphe

4.3. Commandes de modification des caractéristiques de formatage pour les éléments logiques courants

4.3.1. Introduction

Les commandes permettent de modifier les caractéristiques de formage définies par les commandes du niveau global et/ou par le format "standard" ; les modifications portent sur l'élément logique courant. Les commandes se présentent sous deux formes différentes :

- le premier type de commande permet de modifier les caractéristiques de formatage pour tout l'élément logique courant
- le deuxième type de commande permet de modifier les caractéristiques de formatage pour une partie de l'élément logique courant : cette partie est délimitée par des commandes de début et de fin

4.3.2. Description des commandes pour tout l'élément logique courant

1. Format de la commande :

\\<COMMANDE>.

ou

\\<COMMANDE> : <VALEUR>.

2. Fonction :

Ce type de commande permet de modifier les caractéristiques de formatage pour tout l'élément logique courant

3. Liste des commandes :

Séquence : \\CE(ntre).

Effet : l'élément logique courant est centré

Séquence : \\DE(calage) : VALEUR.

Effet : cette commande permet de décaler la première ligne du paragraphe courant ou les éléments d'une liste d'un certain nombre de millimètres ; VALEUR étant ce nombre ; dans le cas d'un paragraphe, cette commande suit immédiatement la commande de création de celui-ci

Séquence : \\MA(rge) : G(auche) : VALEUR.

Effet : cette commande permet de déterminer le décalage en nombre de millimètres par rapport au bord gauche d'une feuille du document pour l'élément logique courant ; VALEUR est ce nombre

Séquence : \\MA(rge) : D(droite) : VALEUR.

Effet : cette commande permet de déterminer le décalage en nombre de millimètres par rapport au bord droit d'une feuille du document pour l'élément logique courant ; VALEUR est ce nombre

Séquence : \\JU(stifie)

Effet : cette commande permet de justifier l'élément logique courant à gauche et à droite

Séquence :

\\cr.
ou
\\LI(gne).

Effet : ces commandes provoquent un saut de ligne des caractéres suivant la commande au sein de l'élément logique courant

Séquence :

\\ff.
ou
\\PA(ge).

Effet : ces commandes provoquent un saut de page de l'élément logique courant

4.3.3. Description des commandes pour une partie de l'élément logique courant

1. Format de la commande :

\\<COMMANDE> : <VALEUR>.

ou

\\<FIN> : <COMMANDE>.

2. Fonction :

Ce type de commande permet de modifier les caractéristiques de formatage pour une partie de l'élément logique courant

3. Liste des commandes :

Séquence : **\\IMP(plémentation) : TYPE.**

Effet : cette commande permet de pré-fixer tous les éléments de la liste par un caractère TYPE ; les différents types disponibles sont le point, le tiret ou l'étoile ; TYPE prend les valeurs :

- **T(iret)**
- **P(oint)**
- **E(toile)**

Séquence : **\\FO(nte) : TYPE.**

Effet : cette commande permet de mettre les caractères suivants dans la fonte spécifiée par VALEUR . Les caractères garderont la même fonte soit jusque la fin du texte, soit jusque la rencontre d'une commande de fin (nb. 1)

Séquence : **\\PO(lice) : type.**

Effet : cette commande permet de mettre les caractères suivants dans la police spécifiée par VALEUR . Les caractères garderont la même police soit jusque la fin du texte, soit jusque la rencontre d'une commande de fin ; (nb. 2)

Séquence : **\\SO(ulignement) : TYPE.**

Effet : ces commandes permettent de définir le type de soulignement. Les caractères garderont le même soulignement soit jusque la fin du texte, soit jusque la rencontre d'une commande de fin (décrite ci-dessous) ; les différents types de soulignement sont le continu, le discontinu ou le supérieur ; TYPE peut prendre les valeurs :

NB: 1 Les différents types de fonte sont donnés au point 6

NB: 2 Les différents types de police sont donnés au point 6

- **C(ontinu)**
- **D(iscontinu)**
- **S(upérieur)**

Séquence : **\\F(in) : NO(te).**

Effet : cette commande détermine que les caractères qui suivent cette commande ne constituent plus une note bas de page

Séquence : **\\F(in) : FO(nte).**

Effet : cette commande définit pour les caractères suivants la fonte pré-définie par les commandes du niveau global ou par le format "standard"

Séquence : **\\F(in) : PO(lice).**

Effet : cette commande définit pour les caractères suivants la police pré-définie par les commandes du niveau global ou par le format "standard"

Séquence : **\\F(in) : SO(ulignement).**

Effet : cette commande définit pour les caractères suivants le type de soulignement pré-défini par les commandes du niveau global ou par le format "standard"

L'utilisateur ne disposant pas d'un clavier qui comprend les lettres accentuées, peut introduire celles-ci de la manière suivante :

- \'e correspond à la lettre é
- \'e correspond à la lettre è
- \^e correspond à la lettre ê
- \"e correspond à la lettre ë
- \'u correspond à la lettre ù
- \"u correspond à la lettre ü
- \^u correspond à la lettre û
- \'a correspond à la lettre à
- \"a correspond à la lettre ä
- \^a correspond à la lettre â
- \"i correspond à la lettre ï
- \^i correspond à la lettre î
- \"o correspond à la lettre ö
- \^o correspond à la lettre ô
- \\ correspond au caractère \

CLASSIFICATION DES COMMANDES PAR ORDRE ALPHABETIQUE

5. CLASSIFICATION DES COMMANDES PAR ORDRE ALPHABETIQUE

5.1. Introduction

Comme il a été signalé dans l'introduction (au point 1), il existe deux types de commandes :

- les commandes du niveau global
- les commandes du niveau local

Ce chapitre donne une classification des commandes par ordre alphabétique et par niveau ; chaque commande est suivie d'une brève définition, les commandes étant explicitées plus en détail dans les chapitres précédents.

5.2. Classification des commandes du niveau global

\\CE(ntre) : AU(teur).
l'élément logique auteur sera centré

\\CE(ntre) : DA(te).
l'élément logique date sera centré

\\CE(ntre) : EN(tête) : (niveau) : 1.
l'élément logique entête de niveau 1 sera centré

\\CE(ntre) : EN(tête) : (niveau) : 2.
l'élément logique entête de niveau 2 sera centré

\\CE(ntre) : EN(tête) : (niveau) : 3.
l'élément logique entête de niveau 3 sera centré

\\CE(ntre) : EN(tête) : (niveau) : 4.
l'élément logique entête de niveau 4 sera centré

\\CE(ntre) : FI(gure).
l'élément logique figure sera centré

\\CE(ntre) : LI(ste).
l'élément logique liste sera centré

\\CE(ntre) : NO(te bas de page).
l'élément logique note bas de page sera centré

\\CE(ntre) : PA(ragraphe).
l'élément logique paragraphe sera centré

\\CE(ntre) : RE(sumé).
l'élément logique résumé sera centré

\\CE(ntre) : TI(tre).
l'élément logique titre sera centré

\\CE(ntre) : TA(ble des matières).
les éléments constituant la table des matières seront centrés

\\CO(lonnage).
le texte sera formaté en double colonne

\\DE(calage) : PA(ragraphe) : VALEUR.
la première ligne de(s) paragraphe(s) sera décalée vers la droite

\\DE(calage) : LI(ste) : VALEUR.
tous les éléments de(s)_liste(s) seront décalés vers la droite

\\EC(art entre éléments logiques) : VALEUR.
spécification de l'espace entre les éléments logiques

\\FO(nte) : AU(teur) : TYPE.
spécification de la fonte pour les éléments logiques de type auteur

\\FO(nte) : DA(te) : TYPE.
spécification de la fonte pour les éléments logiques de type date

\\FO(nte) : EN(tête) : (niveau) : 1 : TYPE.
spécification de la fonte pour les éléments logiques de type entête de niveau 1

\\FO(nte) : EN(tête) : (niveau) : 2 : TYPE.
spécification de la fonte pour les éléments logiques de type entête de niveau 2

\\FO(nte) : EN(tête) : (niveau) : 3 : TYPE.
spécification de la fonte pour les éléments logiques de type entête de niveau 3

\\FO(nte) : EN(tête) : (niveau) : 4 : TYPE.
spécification de la fonte pour les éléments logiques de type entête de niveau 4

\\FO(nte) : FI(gure) : TYPE.
spécification de la fonte pour les éléments logiques de type figure

\\FO(nte) : LI(ste) : TYPE.
spécification de la fonte pour les éléments logiques de type liste

\\FO(nte) : NO(te) bas de page) : TYPE.
spécification de la fonte pour les éléments logiques de type note bas de page

\\FO(nte) : PA(ragraphe) : TYPE.
spécification de la fonte pour les éléments logiques de type paragraphe

\\FO(nte) : RE(sumé) : TYPE.
spécification de la fonte pour l'élément logique de type résumé

\\FO(nte) : TA(ble des matières) : TYPE.
spécification de la fonte pour l'élément logique de type table des matières

\\FO(nte) : TI(tre) : TYPE.
spécification de la fonte pour les éléments logiques de type titre

\\LI(ste) : IMP(lémentation) : TYPE.
type d'implémentation pour les listes

\\IMP(ression réduite) : NUMERO 1 , NUMERO 2.
impression d'un nombre réduit de pages

\\IN(dex).(*)
le texte sera suivi d'un index

\\INT(erligne) : AU(teur) : VALEUR.
spécification de l'interligne pour les éléments logiques de type auteur

\\INT(erligne) : DA(te) : VALEUR.
spécification de l'interligne pour les éléments logiques de type date

\\INT(erligne) : EN(tête) : (niveau) : 1 : VALEUR.
spécification de l'interligne pour les éléments logiques de type entête de niveau 1

\\INT(erligne) : EN(tête) : (niveau) : 2 : VALEUR.
spécification de l'interligne pour les éléments logiques de type entête de niveau 2

\\INT(erligne) : EN(tête) : (niveau) : 3 : VALEUR.
spécification de l'interligne pour les éléments logiques de type entête de niveau 3

\\INT(erligne) : EN(tête) : (niveau) : 4 : VALEUR.
spécification de l'interligne pour les éléments logiques de type entête de niveau 4

\\INT(erligne) : figure : VALEUR.
spécification de l'interligne pour les éléments logiques de type figure

\\INT(erligne) : LI(ste) : VALEUR.
spécification de l'interligne pour les éléments logiques de type liste

\\INT(erligne) : NO(te) bas de page) : VALEUR.
spécification de l'interligne pour les éléments logiques de type note bas de page

\\INT(erligne) : PA(ragraphe) : VALEUR.
spécification de l'interligne pour les éléments logiques de type paragraphe

\\INT(erligne) : RE(sumé) : VALEUR.
spécification de l'interligne pour l'élément logique de type résumé

\\INT(erligne) : TI(tre) : VALEUR.
spécification de l'interligne pour les éléments logiques de type titre

\\INT(erligne) : TA(ble des matières) : VALEUR.
spécification de l'interligne pour les éléments logiques de type table des matières

\\JU(stifie) : AU(teur).
justification des éléments logiques de type auteur

\\JU(stifie) : DA(te).
justification des éléments logiques de type date

\\JU(stifie) : EN(tête) : (niveau) : 1.
justification des éléments logiques de type entête de niveau 1

\\JU(stifie) : EN(tête) : (niveau) : 2.
justification des éléments logiques de type entête de niveau 2

\\JU(stifie) : EN(tête) : (niveau) : 3.
justification des éléments logiques de type entête de niveau 3

\\JU(stifie) : EN(tête) : (niveau) : 4.
justification des éléments logiques de type entête de niveau 4

\\JU(stifie) : FI(gure).
justification des éléments logiques de type figure

\\JU(stifie) : LI(ste).
justification des éléments logiques de type liste

\\JU(stifie) : NO(te bas de page).
justification des éléments logiques de type note bas de page

\\JU(stifie) : PA(ragraphe).
justification des éléments logiques de type paragraphe

\\JU(stifie) : RE(sumé).
justification de l'élément logique de type résumé

\\JU(stifie) : TA(ble des matières).
justification de l'élément logique de type table des matières

\\JU(stifie) : TI(tre).
justification des éléments logiques de type titre

\\MA(rge) : D(roite) : AU(teur) : VALEUR.
spécification de la marge droite pour les éléments logiques de type auteur

\\MA(rge) : D(roite) : DA(te) : VALEUR.
spécification de la marge droite pour les éléments logiques de type date

\\MA(rge) : D(roite) : EN(tête) : (niveau) : 1 : VALEUR.
spécification de la marge droite pour les éléments logiques de type entête de niveau 1

\\MA(rge) : D(roite) : EN(tête) : (niveau) : 2 : VALEUR.
spécification de la marge droite pour les éléments logiques de type entête de niveau 2

\\MA(rge) : D(roite) : EN(tête) : (niveau) : 3 : VALEUR.
spécification de la marge droite pour les éléments logiques de type entête de niveau 3

\\MA(rge) : D(roite) : EN(tête) : (niveau) : 4 : VALEUR.
spécification de la marge droite pour les éléments logiques de type entête de niveau 4

\\MA(rge) : D(roite) : FI(gure) : VALEUR.
spécification de la marge droite pour les éléments logiques de type figure

\\MA(rge) : D(roite) : LI(ste) : VALEUR.
spécification de la marge droite pour les éléments logiques de type liste

\\MA(rge) : D(roite) : NO(te bas de page) : VALEUR.
spécification de la marge droite pour les éléments logiques de type note bas de page

\\MA(rge) : D(roite) : PA(ragraphe) : VALEUR.
spécification de la marge droite pour les éléments logiques de type paragraphe

\\MA(rge) : D(roite) : RE(sumé) : VALEUR.
spécification de la marge droite pour l'élément logique de type résumé

\\MA(rge) : D(roite) : TA(ble des matières) : VALEUR.
spécification de la marge droite pour l'élément logique de type table des matières

\\MA(rge) : D(roite) : TI(tre) : VALEUR.
spécification de la marge droite pour les éléments logiques de type titre

\\MA(rge) : G(auche) : AU(teur) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type auteur

\\MA(rge) : G(auche) : DA(te) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type date

\\MA(rge) : G(auche) : EN(tête) : (niveau) : 1 : VALEUR.
spécification de la marge gauche pour les éléments logiques de type entête de niveau 1

\\MA(rge) : G(auche) : EN(tête) : (niveau) : 2 : VALEUR.
spécification de la marge gauche pour les éléments logiques de type entête de niveau 2

\\MA(rge) : G(auche) : EN(tête) : (niveau) : 3 : VALEUR.
spécification de la marge gauche pour les éléments logiques de type entête de niveau 3

\\MA(rge) : G(auche) : EN(tête) : (niveau) : 4 : VALEUR.
spécification de la marge gauche pour les éléments logiques de type entête de niveau 4

\\MA(rge) : G(auche) : FI(gure) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type figure

\\MA(rge) : G(auche) : LI(ste) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type liste

\\MA(rge) : G(auche) : (NOte bas de page) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type note bas de page

\\MA(rge) : G(auche) : PA(ragraphe) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type paragraphe

\\MA(rge) : G(auche) : RE(sumé) : VALEUR.
spécification de la marge gauche pour l'élément logique de type résumé

\\MA(rge) : G(auche) : TA(ble des matières) : VALEUR.
spécification de la marge gauche pour l'élément logique de type table des matières

\\MA(rge) : G(auche) : TI(tre) : VALEUR.
spécification de la marge gauche pour les éléments logiques de type titre

\\MA(rge) : I(nférieure) : VALEUR.
spécification de la marge inférieure du document

\\MA(rge) : S(upérieure) : VALEUR.
spécification de la marge supérieure du document

\\NU(mero) : E(ntête : niveau :1) : VALEUR.
initialisation du numéro des entêtes de niveau 1

\\NU(mero) : N(ote) : VALEUR.
initialisation du numéro des notes

\\NU(mero) : F(igure) : VALEUR.
initialisation du numéro des figures

\\NU(mero) : P(age) : VALEUR.
initialisation du numéro des pages

\\PA(gination).
pagination du document

\\PA(gination) : I(nférieure).
pagination en bas de page

\\PA(gination) : S(upérieure).
pagination en haut de page

\\PO(lice) : AU(teur) : TYPE.
spécification de la police pour les éléments logiques de type auteur

\\PO(lice) : DA(te) : TYPE.
spécification de la police pour les éléments logiques de type date

\\PO(lice) : EN(tête) : (niveau) : 1 : TYPE.
spécification de la police pour les éléments logiques de type entête de niveau 1

\\PO(lice) : EN(tête) : (niveau) : 2 : TYPE.
spécification de la police pour les éléments logiques de type entête de niveau 2

\\PO(lice) : EN(tête) : (niveau) : 3 : TYPE.
spécification de la police pour les éléments logiques de type entête de niveau 3

\\PO(lice) : EN(tête) : (niveau) : 4 : TYPE.
spécification de la police pour les éléments logiques de type entête de niveau 4

\\PO(lice) : FI(gure) : TYPE.
spécification de la police pour les éléments logiques de type figure

\\PO(lice) : LI(ste) : TYPE.
spécification de la police pour les éléments logiques de type liste

\\PO(lice) : NO(te) bas de page) : TYPE.
spécification de la police pour les éléments logiques de type note bas de page

\\PO(lice) : PA(ragraphe) : TYPE.
spécification de la police pour les éléments logiques de type paragraphe

\\PO(lice) : RE(sumé) : TYPE.
spécification de la police pour l'élément logique de type résumé

\\PO(lice) : TA(bles des matières) : TYPE.
spécification de la police pour l'élément logique de type table des matières

\\PO(lice) : TI(tre) : TYPE.
spécification de la police pour les éléments logiques de type titre

\\UN(ne page).
impression des éléments d'identifications seuls sur la première page

\\SO(ulignement) : AU(teur) : TYPE.
spécification du type de soulignement pour les éléments logiques de type auteur

\\SO(ulignement) : DA(te) : TYPE.
spécification du type de soulignement pour les éléments logiques de type date

\\SO(ulignement) : EN(tête) : (niveau) : 1 : TYPE.
spécification du type de soulignement pour les éléments logiques de type entête de niveau 1

\\SO(ulignement) : EN(tête) : (niveau) : 2 : TYPE.
spécification du type de soulignement pour les éléments logiques de type entête de niveau 2

\\SO(ulignement) : EN(tête) : (niveau) : 3 : TYPE.
spécification du type de soulignement pour les éléments logiques de type entête de niveau 3

\\SO(ulignement) : EN(tête) : (niveau) : 4 : TYPE.
spécification du type de soulignement pour les éléments logiques de type entête de niveau 4

\\SO(ulignement) : FI(gure) : TYPE.
spécification du type de soulignement pour les éléments logiques de type figure

\\SO(ulignement) : LI(ste) : TYPE.
spécification du type de soulignement pour les éléments logiques de type liste

\\SO(ulignement) : NO(te) bas de page) : TYPE.
spécification du type de soulignement pour les éléments logiques de type note bas de page

\\SO(ulignement) : PA(ragraphe) : TYPE.
spécification du type de soulignement pour les éléments logiques de type paragraphe

\\SO(ulignement) : RE(sumé) : TYPE.
spécification du type de soulignement pour l'élément logique de type résumé

\\SO(ulignement) : TA(ble des matières) : TYPE.
spécification du type de soulignement pour l'élément logique de type table des matières

\\SO(ulignement) : TI(tre) : TYPE.
spécification du type de soulignement pour les éléments logiques de type titre

\\TA(ble des matières).
demande d'une table des matières

\\TA(ble des matières) : VALEUR.
spécification du nombre de niveaux de la table des matières

\\TY(pe feuille) : TYPE.
spécification du type de feuille

5.3. Classification des commandes du niveau local

\AU(teur).
création d'un élément logique de type auteur

\\CE(ntre).
l'élément logique courant est centré

\CH(apitre) (: niveau) : 1.
création d'un élément logique de type entête de niveau 1

\CH(apitre) (: niveau) : 2.
création d'un élément logique de type entête de niveau 2

\CH(apitre) (: niveau) : 3.
création d'un élément logique de type entête de niveau 3

\CH(apitre) (: niveau) : 4.
création d'un élément logique de type entête de niveau 4

\\cr.
saut de ligne

\\DE(calage) : VALEUR.
spécification du décalage de la première ligne du paragraphe ou des éléments de la liste

\DA(te).
création de l'élément logique date

\DO(cument).
début du corps du document

\EN(tête).
création d'un élément logique de type entête

\\F(in) : FO(nte).
retour au type de fonte pré-défini pour les caractères suivants

\\F(in) : FO(nte).
fin d'une note bas de page

\\F(in) : PO(lice).
retour au type de police pré-défini pour les caractères suivants

\\F(in) : SO(ulignement).
retour au type de soulignement pré-défini pour les caractères suivants

\\FF.
saut de page

\FI(gure) : X , Y : 'NOM' .(*)
création de l'élément logique de type figure

\\FO(nte) : TYPE.
spécification de la fonte pour les caractères suivants

\\IMP(plémentation) : TYPE.
spécification du type d'implémentation de l'élément logique de type liste

\\JU(stifie).
justification de l'élément logique courant

\\LI(gne). saut de ligne

\LI(ste).
création de l'élément logique de type liste

\\MA(rge) : D(droite) : VALEUR.
spécification de la marge droite pour l'élément logique courant

\\MA(rge) : G(auche) : VALEUR.
spécification de la marge gauche pour l'élément logique courant

\NO(te bas de page).
création de l'élément logique de type note bas de page

\\PA(ge).
saut de page

\PA(ragraphe).
création de l'élément logique de type paragraphe

\\PO(lice) : TYPE.
spécification de la police pour les caractères suivants

\RE(sumé).
création de l'élément logique de type résumé

\\SO(ulignement) : TYPE.
spécification du type de soulignement pour les caractères suivants

\TI(tre).
création de l'élément logique de type titre

DESCRIPTION FONTE - POLICE

6. DESCRIPTION FONTE - POLICE

6.1. Introduction

Cette partie reprend les différentes possibilités de fonte et de police offertes à l'utilisateur. Chaque police est identifiée par un numéro : de 6 à 36 ; ce numéro correspond à la grandeur des caractères qui est exprimée en points typographiques (240 points par pouce). Pour chaque police, trois types de fonte sont disponibles :

- le roman
- le gras
- l'italique

Le formateur travaille en chasse variable c'est-à-dire que pour une police et une fonte déterminée, chaque caractère a ses propres dimensions (par exemple, un 'M' a une largeur et une hauteur plus grande qu'un 'I').

L'utilisateur spécifie un type de police en introduisant la commande décrite dans les parties précédentes suivie d'un numéro de police ; de même, pour spécifier une fonte, la commande sera suivie d'un type de fonte. Les numéros de police et les types de fonte sont décrits à la page suivante ; seuls les numéros et les types soulignés sont disponibles. Si l'utilisateur spécifie une autre fonte que celles disponibles, celle-ci sera convertie en normal ; Si l'utilisateur spécifie une autre police que celles disponibles, celle-ci sera convertie en police numéro 11.

6 point Roman, Bold, and *Italic*.
7 point Roman, Bold, and *Italic*.
8 point Roman, Bold, and *Italic*.
9 point Roman, Bold, and *Italic*.
10 point Roman, Bold, and *Italic*.
11 point Roman, Bold, and *Italic*.
12 point Roman, Bold, and *Italic*.
14 point Roman, Bold, and *Italic*.
16 point Roman, Bold, and *Italic*.
18 point Roman, Bold, and *Italic*.
20 point Roman, Bold, and *Italic*.
22 point Roman, Bold, and *Italic*.
24 point Roman, Bold, and *Italic*.
28 point Roman, Bold, and *Italic*.
36 point Roman, Bold, and *Italic*.

LISTE ET SPECIFICATION DES ERREURS

7. LISTE ET SPECIFICATION DES ERREURS

7.1. Introduction .

Il est fréquent que l'utilisateur rentrant son texte au terminal commette des fautes d'orthographe, des erreurs dans l'emploi des commandes (ce qui entraîne une mauvaise présentation de son texte sur la page ; par exemple le dernier élément d'une liste qui saute de page), ou des erreurs de syntaxe dans l'introduction des commandes de formatage.

En ce qui concerne les fautes d'orthographe et les erreurs de présentation, il lui est aisé de les détecter, de les corriger, de reformater et d'imprimer son texte afin d'avoir une version définitive. Par contre, les erreurs de syntaxe des commandes de formatage lui sont difficilement décelables. La plupart des formateurs ignore la commande utilisée incorrectement et la version imprimée de son texte ne correspond pas à ce qu'il en attendait. Le formateur "FTTL" détecte ces erreurs de syntaxe et envoie un numéro de l'erreur au terminal lors de la phase de formatage. Ce chapitre donne la liste des numéros des erreurs suivis de leur message correspondant entre ". Pour chaque erreur, la signification du message et la conséquence de l'erreur sont également spécifiées. L'utilisateur pourra en outre obtenir plus de détails quant au(x) erreur(s) en faisant la commande PRINT "nom du fichier des erreurs" (cfr. point 8). Les numéros 20 à 199 concernent les erreurs des commandes du niveau global ; les numéros 200 à 300 concernent les erreurs des commandes du niveau local.

7.2. Liste des erreurs

1 "RENCONTRE DE LA FIN DU FICHIER SOURCE"

la fin du fichier source a été détectée ; la phase de formatage s'arrête automatiquement, rien ne sera imprimé

2 "NIVEAU DE CHAPITRE > 4"

introduction d'un chapitre de niveau > 4 ; le formateur ne permettant que 4 niveaux pósibles, le niveau du chapitre sera considéré égal à 4

3 "ERREUR DANS LA CREATION D'UN ORDRE"

la création d'un ordre est incorrecte; cette commande de création est ignorée,cet ordre n'est pas pris en considération

7 "INTRODUCTION DE CARACTERES INCORRECTS EN ENTETE DE FICHIER"

le formatage s'arrête automatiquement

8 "INTRODUCTION DE CARACTERE(S) AU NIVEAU DU CHAPITRE AVANT L'ENTETE"

la commande de création d'un chapitre de quelque niveau qu'il soit est toujours suivie de la commande de création d'un entête ; le(s) caractère(s) précédant cette dernière commande est (ou sont) ignoré(s)

9 "DECALAGE DU PARAGRAPHE SPECIFIE DANS UN ELEMENT D'UN AUTRE TYPE"

la commande de décalage a été réalisée dans un élément logique autre qu'une liste ou un paragraphe ; la commande est ignorée

15 "RENCONTRE DE TROIS \ ALORS QU'ON N'EST PAS DANS UNE LISTE"

les trois \ permettent de définir un élément d'une liste et sont réservés à cet usage ; la commande est ignorée

16 "SEQUENCE LEXICALE NON RECONNUE"

la création d'un élément constituant le corps du document est incorrecte ;cette commande est ignorée, l'élément n'est pas créé

17 "SEQUENCE D'IMPLEMENTATION NON RECONNUE"

la spécification des indications de formatage pour un élément logique est incorrecte ; cette commande est ignorée, l'élément sera formaté selon le format "standard" le décrivant

20 "CARACTERE SPECIAL NON RECONNU"

l'introduction d'une lettre accentuée est incorrecte ; la lettre ne sera pas accentuée

30 "REDEFINITION DE LA FONTE POUR LE TITRE"

le type de fonte défini pour le titre est incorrect ; cette commande de redéfinition est ignorée, le titre aura la fonte définie par le format "standard"

31 "REDEFINITION DE LA FONTE POUR LA DATE"

le type de fonte défini pour la date est incorrect ; cette commande de redéfinition est ignorée, la date aura la fonte définie par le format "standard"

32 "REDEFINITION DE LA FONTE POUR LE RESUME"

le type de fonte défini pour le résumé est incorrect ; cette commande de redéfinition est ignorée, le résumé aura la fonte définie par le format "standard"

33 "REDEFINITION DE LA FONTE POUR L'AUTEUR"

le type de fonte défini pour l'auteur est incorrect ; cette commande de redéfinition est ignorée, l'auteur aura la fonte définie par le format "standard"

34 "REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 1"

le type de fonte défini pour les entêtes de niveau 1 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 1 auront la fonte définie par le format "standard"

35 "REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 2"

le type de fonte défini pour les entêtes de niveau 2 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 2 auront la fonte définie par le format "standard"

36 "REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 3"

le type de fonte défini pour les entêtes de niveau 3 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 3 auront la fonte définie par le format "standard"

37 "REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 4"

le type de fonte défini pour les entêtes de niveau 4 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 4 auront la fonte définie par le format "standard"

38 "REDEFINITION DE LA FONTE POUR LES TEXTES DE FIGURE"

le type de fonte défini pour les textes de figure est incorrect ; cette commande de redéfinition est ignorée, les textes de figure auront la fonte définie par le format "standard"

39 "REDEFINITION DE LA FONTE POUR LES LISTES"

le type de fonte défini pour les listes est incorrect ; cette commande de redéfinition est ignorée, les listes auront la fonte définie par le format "standard"

40 "REDEFINITION DE LA FONTE POUR LES PARAGRAPHES"

le type de fonte défini pour les paragraphes est incorrect ; cette commande de redéfinition est ignorée, les paragraphes auront la fonte définie par le format "standard"

42 "REDEFINITION DE LA FONTE POUR LES NOTES BAS DE PAGE"

le type de fonte défini pour les notes bas de page est incorrect ; cette commande de redéfinition est ignorée, les notes bas de page auront la fonte définie par le format "standard"

43 "REDEFINITION DE LA FONTE POUR LA TABLE DES MATIERES"

le type de fonte défini pour la table des matières est incorrect ; cette commande de redéfinition est ignorée, la table des matières aura la fonte définie par le format "standard"

44 "REDEFINITION DE LA FONTE - ELEMENT INCORRECT"

l'élément pour lequel la fonte est redéfinie est incorrect ; cette commande de redéfinition est ignorée, l'élément aura la fonte définie par le format "standard"

45 "REDEFINITION DE LA POLICE POUR LE TITRE"

le type de police défini pour le titre est incorrect ; cette commande de redéfinition est ignorée, le titre aura la police définie par le format "standard"

46 "REDEFINITION DE LA POLICE POUR LA DATE"

le type de police défini pour la date est incorrect ; cette commande de redéfinition est ignorée, la date aura la police définie par le format "standard"

47 "REDEFINITION DE LA POLICE POUR LE RESUME"

le type de police défini pour le résumé est incorrect ; cette commande de redéfinition est ignorée, le résumé aura la police définie par le format "standard"

48 "REDEFINITION DE LA POLICE POUR L'AUTEUR"

le type de police défini pour l'auteur est incorrect ; cette commande de redéfinition est ignorée, l'auteur aura la police définie par le format "standard"

49 "REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 1"

le type de police défini pour les entêtes de niveau 1 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 1 auront la police définie par le format "standard"

50 "REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 2"

le type de police défini pour les entêtes de niveau 2 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 2 auront la police définie par le format "standard"

51 "REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 3"

le type de police défini pour les entêtes de niveau 3 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 3 auront la police définie par le format "standard"

52 "REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 4"

le type de police défini pour les entêtes de niveau 4 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 4 auront la police définie par le format "standard"

53 "REDEFINITION DE LA POLICE POUR LES TEXTES DE FIGURE"

le type de police défini pour les textes de figure est incorrect ; cette commande de redéfinition est ignorée, les textes de figure auront la police définie par le format "standard"

54 "REDEFINITION DE LA POLICE POUR LES LISTES"

le type de police défini pour les listes est incorrect ; cette commande de redéfinition est ignorée, les listes auront la police définie par le format "standard"

55 "REDEFINITION DE LA POLICE POUR LES PARAGRAPHES"

le type de police défini pour les paragraphes est incorrect ; cette commande de redéfinition est ignorée, les paragraphes auront la police définie par le format "standard"

56 "REDEFINITION DE LA POLICE POUR LES NOTES BAS DE PAGE"

le type de police défini pour les notes bas de page est incorrect ; cette commande de redéfinition est ignorée, les notes bas de page auront la police définie par le format "standard"

57 "REDEFINITION DE LA POLICE POUR LA TABLE DES MATIERES"

le type de police défini pour la table des matières est incorrect ; cette commande de redéfinition est ignorée, la table des matières aura la police définie par le format "standard"

58 "REDEFINITION DE LA POLICE - ELEMENT NON RECONNU"

l'élément pour lequel la police a été spécifiée est incorrect ; cette commande de redéfinition est ignorée, l'élément aura la police définie dans le format "standard"

59 "REDEFINITION DU FAIT QU'UN ELEMENT DOIT ETRE CENTRE"

l'élément pour lequel le centrage a été demandé est incorrect ; cette commande est ignorée, l'élément sera justifié ou centré suivant la définition du format "standard"

60 "REDEFINITION DU FAIT QU'UN ELEMENT DOIT ETRE JUSTIFIE"

l'élément pour lequel la justification a été demandée est incorrect ; cette commande est ignorée, l'élément sera justifié ou centré suivant la définition du format "standard"

61 "REDEFINITION DE L'INTERLIGNE POUR LE TITRE"

la valeur de l'interligne définie pour le titre est incorrecte ; cette commande de redéfinition est ignorée, le titre aura l'interligne défini par le format "standard"

62 "REDEFINITION DE L'INTERLIGNE POUR LA DATE"

la valeur de l'interligne définie pour la date est incorrecte ; cette commande de redéfinition est ignorée, la date aura l'interligne défini par le format "standard"

63 "REDEFINITION DE L'INTERLIGNE POUR LE RESUME"

la valeur de l'interligne définie pour le résumé est incorrecte ; cette commande de redéfinition est ignorée, le résumé aura l'interligne défini par le format "standard"

64 "REDEFINITION DE L'INTERLIGNE POUR L'AUTEUR"

la valeur de l'interligne définie pour l'auteur est incorrecte ; cette commande de redéfinition est ignorée, l'auteur aura l'interligne défini par le format "standard"

65 "REDEFINITION DE L'INTERLIGNE POUR LES ENTETES DE NIVEAU 1"

la valeur de l'interligne définie pour les entêtes de niveau 1 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 1 auront l'interligne défini par le format "standard"

66 "REDEFINITION DE L'INTERLIGNE POUR LES ENTETES DE NIVEAU 2"

la valeur de l'interligne définie pour les entêtes de niveau 2 est incorrecte ; cette commande de redéfinition est ignorée, les entêtes de niveau 2 auront l'interligne défini par le format "standard"

67 "REDEFINITION DE L'INTERLIGNE POUR LES ENTETES DE NIVEAU 3"

la valeur de l'interligne définie pour les entêtes de niveau 3 est incorrecte ; cette commande de redéfinition est ignorée, les entêtes de niveau 3 auront l'interligne défini par le format "standard"

68 "REDEFINITION DE L'INTERLIGNE POUR LES ENTETES DE NIVEAU 4"

la valeur de l'interligne définie pour les entêtes de niveau 4 est incorrecte ; cette commande de redéfinition est ignorée, les entêtes de niveau 4 auront l'interligne défini par le format "standard"

69 "REDEFINITION DE L'INTERLIGNE POUR LES TEXTES DE FIGURE"

la valeur de l'interligne définie pour les textes de figure est incorrecte ; cette commande de redéfinition est ignorée, les textes de figure auront l'interligne défini par le format "standard"

70 "REDEFINITION DE L'INTERLIGNE POUR LES LISTES"

la valeur de l'interligne définie pour les listes est incorrecte ; cette commande de redéfinition est ignorée, les listes auront l'interligne défini par le format "standard"

71 "REDEFINITION DE L'INTERLIGNE POUR LES PARAGRAPHES"

la valeur de l'interligne définie pour les paragraphes est incorrecte ; cette commande de redéfinition est ignorée, les paragraphes auront l'interligne défini par le format "standard"

72 "REDEFINITION DE L'INTERLIGNE POUR LES NOTES BAS DE PAGE"

la valeur de l'interligne définie pour les notes bas de page est incorrecte ; cette commande de redéfinition est ignorée, les notes bas de page auront l'interligne défini par le format "standard"

73 "REDEFINITION DE L'INTERLIGNE POUR LA TABLE DES MATIERES"

la valeur de l'interligne définie pour la table des matières est incorrecte ; cette commande de redéfinition est ignorée, la table des matières aura l'interligne défini par le format "standard"

74 "REDEFINITION DE L'INTERLIGNE - ELEMENT NON RECONNU"

l'élément pour lequel l'interligne est redéfini est incorrect ; cette commande de redéfinition est ignorée, l'élément aura l'interligne défini par le format "standard"

75 "REDEFINITION DU SOULIGNEMENT POUR LE TITRE"

la valeur du soulignement défini pour le titre est incorrect ; cette commande de redéfinition est ignorée, le titre aura le soulignement défini par le format "standard"

76 "REDEFINITION DU SOULIGNEMENT POUR LA DATE"

le type de soulignement défini pour la date est incorrecte ; cette commande de redéfinition est ignorée, la date aura le soulignement défini par le format "standard"

77 "REDEFINITION DU SOULIGNEMENT POUR LE RESUME"

le type de soulignement défini pour le résumé est incorrect ; cette commande de redéfinition est ignorée, le résumé aura le soulignement défini par le format "standard"

78 "REDEFINITION DU SOULIGNEMENT POUR L'AUTEUR"

le type de soulignement défini pour l'auteur est incorrect ; cette commande de redéfinition est ignorée, l'auteur aura le soulignement défini par le format "standard"

79 "REDEFINITION DU SOULIGNEMENT POUR LES ENTETES DE NIVEAU 1"

le type de soulignement défini pour les entêtes de niveau 1 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 1 auront le soulignement défini par le format "standard"

80 "REDEFINITION DU SOULIGNEMENT POUR LES ENTETES DE NIVEAU 2"

le type de soulignement défini pour les entêtes de niveau 2 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 2 auront le soulignement défini par le format "standard"

81 "REDEFINITION DU SOULIGNEMENT POUR LES ENTETES DE NIVEAU 3"

le type de soulignement défini pour les entêtes de niveau 3 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 3 auront le soulignement défini par le format "standard"

82 "REDEFINITION DU SOULIGNEMENT POUR LES ENTETES DE NIVEAU 4"

le type de soulignement défini pour les entêtes de niveau 4 est incorrect ; cette commande de redéfinition est ignorée, les entêtes de niveau 4 auront le soulignement défini par le format "standard"

83 "REDEFINITION DU SOULIGNEMENT POUR LES TEXTES DE FIGURE"

le type de soulignement défini pour les textes de figure est incorrect ; cette commande de redéfinition est ignorée, les textes de figure auront le soulignement défini par le format "standard"

84 "REDEFINITION DU SOULIGNEMENT POUR LES LISTES"

le type de soulignement défini pour les listes est incorrect ; cette commande de redéfinition est ignorée, les listes auront le soulignement défini par le format "standard"

85 "REDEFINITION DU SOULIGNEMENT POUR LES PARAGRAPHES"

le type de soulignement défini pour les paragraphes est incorrect ; cette commande de redéfinition est ignorée, les paragraphes auront le soulignement défini par le format "standard"

86 "REDEFINITION DU SOULIGNEMENT POUR LES NOTES BAS DE PAGE"

le type de soulignement défini pour les notes bas de page est incorrect ; cette commande de redéfinition est ignorée, les notes bas de page auront le soulignement défini par le format "standard"

87 "REDEFINITION DU SOULIGNEMENT POUR LA TABLE DES MATIERES"

le type de soulignement défini pour la table des matières est incorrect ; cette commande de redéfinition est ignorée, la table des matières aura le soulignement défini par le format "standard"

88 "REDEFINITION DU SOULIGNEMENT - ELEMENT NON RECONNU"

l'élément pour lequel le soulignement est redéfini est incorrect ; cette commande de redéfinition est ignorée, l'élément aura le soulignement défini par le format "standard"

91 "REDEFINITION DE LA MARGE GAUCHE POUR LA DATE"

la valeur de la marge gauche définie pour la date est incorrecte ; cette commande de redéfinition est ignorée, la date aura la marge gauche définie par le format "standard"

92 "REDEFINITION DE LA MARGE GAUCHE POUR LE RESUME"

la valeur de la marge gauche définie pour le résumé est incorrecte ; cette commande de redéfinition est ignorée, le résumé aura la marge gauche définie par le format "standard"

93 "REDEFINITION DE LA MARGE GAUCHE POUR L'AUTEUR"

la valeur de la marge gauche définie pour l'auteur est incorrecte ; cette commande de redéfinition est ignorée, l'auteur aura la marge gauche définie par le format "standard"

94 "REDEFINITION DE LA MARGE GAUCHE POUR LES ENTETES DE NIVEAU 1"

la valeur de la marge gauche définie pour les entêtes de niveau 1 est incorrect e; cette commande de redéfinition est ignorée, les entêtes de niveau 1 auront la marge gauche définie par le format "standard"

95 "REDEFINITION DE LA MARGE GAUCHE POUR LES ENTETES DE NIVEAU 2"

la valeur de la marge gauche définie pour les entêtes de niveau 2 est incorrect e; cette commande de redéfinition est ignorée, les entêtes de niveau 2 auront la marge gauche définie par le format "standard"

96 "REDEFINITION DE LA MARGE GAUCHE POUR LES ENTETES DE NIVEAU 3"

la valeur de la marge gauche définie pour les entêtes de niveau 3 est incorrect e; cette commande de redéfinition est ignorée, les entêtes de niveau 3 auront la marge gauche définie par le format "standard"

97 "REDEFINITION DE LA MARGE GAUCHE POUR LES ENTETES DE NIVEAU 4"

la valeur de la marge gauche définie pour les entêtes de niveau 4 est incorrecte ; cette commande de redéfinition est ignorée, les entêtes de niveau 4 auront la marge gauche définie par le format "standard"

98 "REDEFINITION DE LA MARGE GAUCHE POUR LES TEXTES DE FIGURE"

la valeur de la marge gauche définie pour les textes de figure est incorrecte ; cette commande de redéfinition est ignorée, les textes de figure auront la marge gauche définie par le format "standard"

99 "REDEFINITION DE LA MARGE GAUCHE POUR LES LISTES"

la valeur de la marge gauche définie pour les listes est incorrecte ; cette commande de redéfinition est ignorée, les listes auront la marge gauche définie par le format "standard"

100 "REDEFINITION DE LA MARGE GAUCHE POUR LES PARAGRAPHES"

la valeur de la marge gauche définie pour les paragraphes est incorrecte ; cette commande de redéfinition est ignorée, les paragraphes auront la marge gauche définie par le format "standard"

101 "REDEFINITION DE LA MARGE GAUCHE POUR LES NOTES BAS DE PAGE"

la valeur de la marge gauche définie pour les notes bas de page est incorrecte ; cette commande de redéfinition est ignorée, les notes bas de page auront la marge gauche définie par le format "standard"

102 "REDEFINITION DE LA MARGE DROITE POUR LE TITRE"

la valeur de la marge droite définie pour le titre est incorrecte ; cette commande de redéfinition est ignorée, le titre aura la marge droite définie par le format "standard"

103 "REDEFINITION DE LA MARGE DROITE POUR LA DATE"

la valeur de la marge droite définie pour la date est incorrecte ; cette commande de redéfinition est ignorée, la date aura la marge droite définie par le format "standard"

104 "REDEFINITION DE LA MARGE DROITE POUR LE RESUME"

la valeur de la marge droite définie pour le résumé est incorrecte ; cette commande de redéfinition est ignorée, le résumé aura la marge droite définie par le format "standard"

105 "REDEFINITION DE LA MARGE DROITE POUR L'AUTEUR"

la valeur de la marge droite définie pour l'auteur est incorrecte ; cette commande de redéfinition est ignorée, l'auteur aura la marge droite définie par le format "standard"

106 "REDEFINITION DE LA MARGE DROITE POUR LES ENTETES DE NIVEAU 1"

la valeur de la marge droite définie pour les entêtes de niveau 1 est incorrecte; ecette commande de redéfinition est ignorée, les entêtes de niveau 1 auront la marge droite définie par le format "standard"

107 "REDEFINITION DE LA MARGE DROITE POUR LES ENTETES DE NIVEAU 2"

la valeur de la marge droite définie pour les entêtes de niveau 2 est incorrecte; cette commande de redéfinition est ignorée, les entêtes de niveau 2 auront la marge droite définie par le format "standard"

108 "REDEFINITION DE LA MARGE DROITE POUR LES ENTETES DE NIVEAU 3"

la valeur de la marge droite définie pour les entêtes de niveau 3 est incorrecte ; cette commande de redéfinition est ignorée, les entêtes de niveau 3 auront la marge droite définie par le format "standard"

109 "REDEFINITION DE LA MARGE DROITE POUR LES ENTETES DE NIVEAU 4"

la valeur de la marge droite définie pour les entêtes de niveau 4 est incorrecte ; cette commande de redéfinition est ignorée, les entêtes de niveau 4 auront la marge droite définie par le format "standard"

110 "REDEFINITION DE LA MARGE DROITE POUR LES TEXTES DE FIGURE"

la valeur de la marge droite définie pour les textes de figure est incorrecte ; cette commande de redéfinition est ignorée, les textes de figure auront la marge droite définie par le format "standard"

111 "REDEFINITION DE LA MARGE DROITE POUR LES LISTES"

la valeur de la marge droite définie pour les listes est incorrecte ; cette commande de redéfinition est ignorée, les listes auront la marge droite définie par le format "standard"

112 "REDEFINITION DE LA MARGE DROITE POUR LES PARAGRAPHES"

la valeur de la marge droite définie pour les paragraphes est incorrecte ; cette commande de redéfinition est ignorée, les paragraphes auront la marge droite définie par le format "standard"

113 "REDEFINITION DE LA MARGE DROITE POUR LES NOTES BAS DE PAGE"

la valeur de la marge droite définie pour les notes bas de page est incorrecte ; cette commande de redéfinition est ignorée, les notes bas de page auront la marge droite définie par le format "standard"

114 "MARGE GAUCHE/MARGE DROITE DU TITRE TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

115 "MARGE GAUCHE/MARGE DROITE DE LA DATE TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

116 "MARGE GAUCHE/MARGE DROITE DU RESUME TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

117 "MARGE GAUCHE/MARGE DROITE DE L'AUTEUR TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

118 "MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 1 TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

119 "MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 2 TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

120 "MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 3 TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

121 "MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 4 TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

122 "MARGE GAUCHE/MARGE DROITE DES TEXTES DE FIGURES TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

123 "MARGE GAUCHE/MARGE DROITE DES LISTES TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

124 "MARGE GAUCHE/MARGE DROITE DES NOTES BAS DE PAGE TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

125 "MARGE GAUCHE/MARGE DROITE DES PARAGRAPHES TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

126 "MARGE GAUCHE/MARGE DROITE DE LA TABLE DES MATIERES TROP GRANDE"

la valeur de redéfinition de la marge gauche/marge droite est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

127 "MARGE SUPERIEURE/MARGE INFERIEURE DE LA TABLE DES MATIERES TROP GRANDE"

la valeur de redéfinition de la marge supérieure/marge inférieure du document est trop grande ; cette commande de redéfinition est ignorée, la marge sera celle définie par le format "standard"

128 "REDEFINITION DE LA MARGE GAUCHE - ELEMENT NON RECONNU"

l'élément pour lequel la marge gauche est redéfinie est incorrect ; cette commande de redéfinition est ignorée, l'élément aura la marge gauche définie par le format "standard"

129 "REDEFINITION DE LA MARGE DROITE - ELEMENT NON RECONNU"

l'élément pour lequel la marge droite est redéfinie est incorrect ; cette commande de redéfinition est ignorée, l'élément aura la marge droite définie par le format "standard"

130 "REDEFINITION DE LA MARGE GAUCHE POUR LA TABLE DES MATIERES"

la valeur de la marge gauche définie pour la table des matières est incorrecte ; cette commande de redéfinition est ignorée, la table des matières aura la marge gauche définie par le format "standard"

131 "REDEFINITION DE LA MARGE DROITE POUR LA TABLE DES MATIERES"

la valeur de la marge droite définie pour la table des matières est incorrecte ; cette commande de redéfinition est ignorée, la table des matières aura la marge droite définie par le format "standard"

132 "INITIALISATION DU NUMERO DES ENTETES DE NIVEAU 1"

la spécification du numéro de la première entête de niveau 1 est incorrect ; cette commande de redéfinition est ignorée, le numéro sera initialisé à 1

133 "INITIALISATION DU NUMERO DES PAGES"

la spécification du numéro de la première page est incorrecte ; cette commande de redéfinition est ignorée, le numéro sera initialisé à 1

134 "REDEFINITION DU DECALAGE DE LA PREMIERE LIGNE DES PARAGRAPHES"

la valeur du décalage définie pour la première ligne des paragraphes est incorrecte ; cette commande de redéfinition est ignorée, le décalage aura la valeur définie par le format "standard"

135 "REDEFINITION DU DECALAGE DES LISTES"

la valeur du décalage définie pour les listes est incorrecte ; cette commande de redéfinition est ignorée, le décalage aura la valeur définie par le format "standard"

136 "REDEFINITION DU NOMBRE DE MILLIMETRES ENTRE ELEMENTS"

la valeur de l'espace entre les éléments est incorrecte ; cette commande de redéfinition est ignorée, l'espace aura la valeur définie par le format "standard"

137 "REDEFINITION DU DECALAGE - SPECIFICATION DU TYPE D'ELEMENT INCONNU"

l'élément pour lequel le décalage est redéfini est incorrect ; cette commande de redéfinition est ignorée, l'élément aura le décalage défini par le format "standard"

138 "INITIALISATION DU NUMERO DES FIGURES"

la spécification du numéro de la première figure est incorrecte ; cette commande de redéfinition est ignorée, le numéro sera initialisé à 1

139 "INITIALISATION DU NUMERO DES NOTES BAS DE PAGE"

la spécification du numéro de la première note est incorrecte ; cette commande de redéfinition est ignorée, le numéro sera initialisé à 1

140 "INITIALISATION DU NUMERO - SPECIFICATION DU TYPE D'ELEMENT INCONNU"

l'élément pour lequel le numéro est initialisé est incorrect ; cette commande de redéfinition est ignorée, l'élément aura le numéro initialisé à 1

141 "REDEFINITION DU TYPE D'IMPLEMENTATION DES LISTES"

l'implémentation demandée est incorrecte ; cette commande de redéfinition est ignorée, L'implémentation aura la valeur définie par le format "standard"

142 "DEFINITION DU NOMBRE DE NIVEAUX DE LA TABLE DES MATIERES"

le nombre de niveaux pour la table des matières est incorrect ; cette commande de redéfinition est ignorée, le nombre de niveaux sera automatiquement fixé à 4

142 "REDEFINITION NON RECONNUE"

erreur en général d'une commande de modification des indications de formatage ; cette commande est ignorée ; les indications de formatage resteront spécifiées par le format "standard"

145 "IMPRESSION REDUITE"

le nombre de pages demandé à être imprimées est incorrect ; cette commande de redéfinition est ignorée, toutes les pages seront imprimées

181 "REDEFINITION DE LA MARGE DROITE POUR LA TABLE DES MATIERES"

la valeur de la marge droite définie pour la table des matières est incorrecte ; cette commande de redéfinition est ignorée, la table des matières aura la marge droite définie par le format "standard"

182 "REDEFINITION DE LA MARGE SUPERIEURE"

la valeur de la marge supérieure définie est incorrecte ; cette commande de redéfinition est ignorée, la marge supérieure aura la valeur définie par le format "standard"

183 "REDEFINITION DE LA MARGE INFERIEURE"

la valeur de la marge inférieure définie est incorrecte ; cette commande de redéfinition est ignorée, la marge inférieure aura la valeur définie par le format "standard"

200 "RENCONTRE D'UN CHAPITRE DE NIVEAU 2 INCORRECT"

spécification de la création d'un chapitre de niveau 2 alors qu'il n'existe pas de chapitre de niveau 1 ; cette commande de création est ignorée, ce chapitre n'est pas créé

201 "RENCONTRE D'UN CHAPITRE DE NIVEAU 3 INCORRECT"

spécification de la création d'un chapitre de niveau 3 alors qu'il n'existe pas de chapitre de niveau 2 ; cette commande de création est ignorée, ce chapitre n'est pas créé

202 "RENCONTRE D'UN CHAPITRE DE NIVEAU 4 INCORRECT"

spécification de la création d'un chapitre de niveau 4 alors qu'il n'existe pas de chapitre de niveau 3 ; cette commande de création est ignorée, ce chapitre n'est pas créé

203 "RENCONTRE DE "\\C_" OU "_" N'EST PAS DEFINI CORRECTEMENT"

les caractères introduits pour spécifier la commande sont insuffisants pour son identification ; cette commande est ignorée

204 "ERREUR DANS LA SPECIFICATION DE LA POLICE"

la police est spécifiée incorrectement ; la commande est ignorée, les caractères sur lesquels portait ce changement garderont la police définie par le format "standard" ou par des commandes du niveau global

206 "RENCONTRE DE "\\P_" OU "_" N'EST PAS DEFINI CORRECTEMENT"

les caractères introduits pour spécifier la commande sont insuffisants pour son identification ; cette commande est ignorée

207 "ERREUR DANS LA SPECIFICATION DE LA FONTE"

la fonte est spécifiée incorrectement ; la commande est ignorée, les caractères sur lesquels portait ce changement garderont la fonte définie par le format "standard" ou par des commandes du niveau global

208 "ERREUR DANS LE CODE DE FIN DE SOULIGNEMENT SOIT DE POLICE SOIT DE FONTE"

la commande de fin de soulignement, de police ou de fonte est incorrecte ; les caractères suivant cette commande garderont le soulignement, la fonte ou la police qui ont été spécifiés jusqu'à la rencontre d'une commande de création d'un élément suivant ou d'une commande de redéfinition du soulignement, de la police ou de la fonte

209 "RENCONTRE DE "\\F_" OU "_" N'EST PAS DEFINI CORRECTEMENT"

les caractères introduits pour spécifier la commande sont insuffisants pour son identification ; cette commande est ignorée

210 "ERREUR DANS LA SPECIFICATION D'IMPLEMENTATION D'UNE LISTE"

l'implémentation demandée est incorrecte ; cette commande de redéfinition est ignorée, L'implémentation aura la valeur définie par le format "standard" ou par les commandes de niveau global

211 "RENCONTRE DE "\\L_" OU "_" N'EST PAS DEFINI CORRECTEMENT"

les caractères introduits pour spécifier la commande sont insuffisants pour son identification ; cette commande est ignorée

212 "ERREUR DANS LA SPECIFICATION DU TYPE DE SOULIGNEMENT"

le type de soulignement est spécifié incorrectement ; la commande est ignorée, les caractères sur lesquels portait ce changement garderont le type de soulignement défini par le format "standard" ou par des commandes du niveau global

213 "ERREUR DANS LA SPECIFICATION DU DECALAGE D'UN PARAGRAPHE"

la valeur du décalage définie pour la première ligne des paragraphes est incorrecte ; cette commande de redéfinition est ignorée, le décalage aura la valeur définie par le format "standard" ou par les commandes du niveau global

214 "ERREUR DANS LA SPECIFICATION DE LA MARGE GAUCHE"

la valeur de la marge gauche définie pour l'élément est incorrecte ; cette commande de redéfinition est ignorée, l'élément aura la marge gauche définie par le format "standard" ou par les commandes du niveau global

215 "ERREUR DANS LA SPECIFICATION DE LA MARGE DROITE"

la valeur de la marge droite définie pour l'élément est incorrecte ; cette commande de redéfinition est ignorée, l'élément aura la marge droite définie par le format "standard" ou par les commandes du niveau global

216 "TYPE DE MARGE NON RECONNUE"

les caractères introduits pour spécifier la commande de modification d'une marge sont insuffisants pour son identification ; cette commande est ignorée, aucune marge n'est modifiée

217 "ERREUR DANS UNE COMMANDE D'IMPLEMENTATION"

erreur en général d'une commande de modification des indications de formatage ; cette commande est ignorée ; les indications de formatage resteront spécifiées par le format "standard" ou par les commandes du niveau global

220 "ERREUR DANS LA SPECIFICATION DU DECALAGE D'UNE LISTE"

la valeur du décalage définie pour les éléments d'une liste est incorrecte ; cette commande de redéfinition est ignorée, les éléments auront le décalage défini par le format "standard" ou par les commandes du niveau global

300 "INTRODUCTION D'UNE CHAINE DE CARACTERES TROP LONGUE"

une chaine de caractères est composée de plus de 24 caractères sans aucun caractère "blanc" ; les caractères à partir du 24 ième sont ignorés

EXEMPLE D'UTILISATION

8. EXEMPLE D'UTILISATION

8.1. Introduction

Ce chapitre comprendra trois parties :

- les commandes pour l'utilisation du formateur
- un "bon exemple" de texte formaté avec le texte au kilomètre
- un "mauvais exemple" de texte formaté avec le texte au kilomètre

L'utilisateur pourra se référer au résumé de l'exemple pour avoir des indications sur les commandes de formatage.

remarque : le numéro de page encerclé a été rajouté
et correspond à la pagination du manuel

8.2. Commandes pour l'utilisation du formateur

Pour réaliser le formatage, l'utilisateur dispose de deux commandes suivant qu'il désire ou non utiliser des macros.

Si l'utilisateur désire utiliser des macros, il utilisera la commande :

@FTTL <FICHIER 1> <FICHIER 2> <FICHIER 3> <FICHIER 4>

où

<FICHIER 1> : nom du fichier contenant le texte à formater

<FICHIER 2> : nom du fichier contenant les macros

<FICHIER 3> : nom du fichier qui contiendra le résultat du formatage

<FICHIER 4> : nom du fichier qui contiendra les erreurs

Si l'utilisateur désire **ne pas** utiliser des macros, il utilisera la commande :

@FTTL <FICHIER 1> <FICHIER 2> <FICHIER 3>

où

<FICHIER 1> : nom du fichier contenant le texte à formater

<FICHIER 2> : nom du fichier qui contiendra le résultat du formatage

<FICHIER 3> : nom du fichier qui contiendra les erreurs

L'extension par défaut d'un fichier sera **.DAT**

8.3. Exemples de texte formaté

Chaque exemple reprend le document formaté et imprimé sur l'imprimante Laser suivi de leur texte source (texte au kilomètre).

8.3.1. "BON" EXEMPLE

Mathieu - Schmitz & David Levy

Lausanne, 12 - avril - 1983

Le texte source se compose de caractères constituant les commandes du niveau global, les commandes du niveau local et le texte.
Le contenu du texte n'a aucune importance. Les commandes du niveau global sont les commandes situées avant la séquence <\DOCU.>. Elles ont été choisies afin de présenter les différentes commandes de formatage et afin de présenter un texte "propre". Des commandes de niveau global et des commandes de niveau local seront présentes dans ce texte. Elles seront présentées en majuscule et sous leur forme réduite, ceci n'étant en rien une contrainte.

1. INTRODUCTION

Je vous présente aujourd'hui un jeu que vous n'avez probablement **jamais** rencontré. Cousin des échecs, sa complexité est plus grande d'où des parties passionnantes qui se poursuivent sur un nombre de coups nettememt supérieur à celui des échecs. Ce jeu est si populaire au Japon que les meilleurs deviennent souvent millionnaires et sont plus vénérés que Bjorn Borg en Suède ou Keeving Keegan en Angleterre. Je conseille à tous ceux qui aiment les échecs "occidentaux" de tâter du shogi, les échecs japonais.

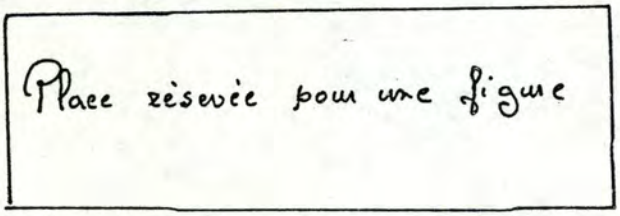
2. INITIATION

Ceux de mes amis qui jouent au shogi le trouvent passionnant et d'autres apprecieraient le jeu après une heure ou deux d'apprentissage. Si le Japon est la "Mecque" du shogi, une organisation s'est développée dans le monde occidental durant ces dernières années ; son but : populariser le jeu hors de son pays natal. Tâche qui s'avère difficile étant donné la non-popularité de cette discipline et la concurrence que lui fait le traditionnel jeu d'echecs qui, lui, est répandu dans le monde entier et se popularise grâce à des concours internationaux et à l'affrontement des géants de ce jeu. L'association de shogi accueille les nouveaux membres et fournit des jeux de shogi et des livres d'initiation à ceux qui ne peuvent les trouver ailleurs. Elle publie également un magazine et propose des rencontres à Londres.

Grâce à ces efforts, des tournois de shogi ont eu lieu un peu partout dans le monde. Mon collègue americain Larry Kauffman, maître international d'échecs, est devenu complètement fou de ce jeu; il s'est même rendu au Japon dans l'espoir de devenir un grand champion de cette discipline.

2.1. Comment jouer au shogi?

Chaque joueur entame la partie avec 20 pièces en bois ou en plastique. Ces pièces sont de la même couleur. Le jeu se déroule sur un échiquier 9 * 9 (connaît-on un microprocesseur à 9 bits) avec deux armées situées dans la position suivante (fig. 1)



- Fig. 1 - la place est réservée sur la page pour insérer une figure de nom spécifié par l'utilisateur ; ceci constitue la légende, une référence est insérée dans le texte à l'endroit de son "appel"

La meilleure façon d'apprendre est d'acheter le livre de FAIRBANK avec un guide près de vous (nb. 1) . Les jeux se composent de pièces marquées avec des caractères japonais mais sont importé s avec les pièces marquées uniquement de caractères occidentaux et des flèches qui vous indiquent leur sens de déplacement. L'apprentissage du shogi n'est pas plus long que celui des échecs, et je me suis laissé dire de source sûre que l'on s'accoutume aux symboles japonais bien plus vite que l'on ne le penserait. Le but de cet article étant de permettre au lecteur d'écrire son propre programme de shogi, je commencerai donc par résumer les règles et les coups du jeu.

2.2. Les pièces et leur déplacement

Le roi : chaque joueur a un roi et comme aux échecs, le but du jeu est de mettre échec et mat le roi adverse. Comme aux échecs, le roi peut se déplacer d'une case dans toutes les directions (horizontale, verticale ou diagonale).

Le général d'or : en début de partie chaque joueur dispose de deux de ces pièces qui avancent d'une case à la fois à la verticale, à l'horizontale et en diagonale vers l'avant. Ils ne peuvent pas se déplacer en diagonale vers l'arrière.

Le général d'argent : chaque joueur dispose également de deux de ces pièces. Celles-ci avancent d'une case en diagonale ou vers l'avant. Elles ne peuvent ni aller sur le côté ni reculer directement.

Le cavalier : chaque joueur ã deux cavaliers dont le déplacement est semblable à celui des échecs mais avec la restriction suivante : ils ne peuvent avancer que de deux cases puis aller sur la case à gauche ou à droite, alors qu'aux échecs le cavalier a huit coups à sa disposition depuis une case centrale sur un échiquier vide, un cavalier de shogi n'aura que deux possibilités, mais il peut sauter comme aux échecs.

Le lancier : chacun des deux lanciers avance droit devant lui autant qu'il le desire, mais ils ne peuvent en aucun cas sauter une pièce de l'adversaire.

NB: 1 ce livre s'intitule : "How to play Shogi"

La tour : la tour de shogi se déplace exactement comme son homologue des échecs c'est-à-dire en ligne droite, d'autant de cases qu'elle le désire. Comme il n'y a pas de dame au shogi, on considère généralement que la tour est la pièce la plus puissante de toutes celles de l'échiquier.

Le fou : cette pièce se déplace également comme un fou dans le jeu d'échec n'importe quel nombre de cases en diagonale.

Le pion : comme aux échecs, la position initiale de shogi comporte une rangée de pions qui barre la largeur de l'échiquier. Chaque joueur commence la partie avec neuf pions qui se déplacent vers l'avant. Il n'y a pas de déplacement double lors du premier coup ; il n'y a pas de prise en diagonale et il n'y a pas de prise en passant.
Pour résumer voici une liste des pièces :

- le pion
- le cavalier
- la tour
- le lancier
- le général d'or
- le général d'argent
- le roi

2.3. Attention aux pièces promues

Tandis qu'aux échecs seuls les pions peuvent être promus et deviennent des pièces de valeur supérieure, l'interêt du shogi provient de ce que certaines autres pièces peuvent également être promues. On réalise un coup de promotion en déplacant en partie ou complètement dans la zone de promotion de l'adversaire (les trois dernières rangées les plus éloignées de vous). La promotion a lieu à la fin d'un tel coup mais il n'est pas toujours obligatoire de promouvoir au shogi. Plusieurs pièces peuvent être promues.

2.3.1. Cas du général d'argent

Une fois promu, il se déplace comme un général d'or. Sur votre jeu de shogi, le général d'argent peut être retourné et sur l'autre coté, vous verrez le symbole d'un général d'argent qui vient d'être promu.

2.3.1.1. Cas des autres pièces

Le cavalier, le lancier, le pion se déplacent également comme le général d'or une fois qu'ils viennent d'être promus.
La tour lorsqu'elle est promue garde sa capacité originelle de déplacement d'un nombre quelconque de cases à l'horizontale ou à la verticale mais elle acquiert la possibilité de se déplacer d'une case en diagonale. Le fou, de la même manière acquiert la possibilité de se déplacer d'une case supplémentaire à l'horizontale ou à la verticale. Si un pion ou un lancier se déplace jusqu'au dernier rang ou si un cavalier atteint un des deux premiers rangs, la promotion est obligatoire. Dans tous les autres cas la promotion est optionnelle. Il vaut mieux en général promouvoir ; cela assure un avantage certain sur l'adversaire à condition qu'il n'ait déjà pas eu l'occasion de promouvoir.

3. UNE PIECE PRISE AU JEU PEUT ETRE REMISE EN JEU

Si un joueur déplace une de ses pièces sur une case occupée par l'adversaire, cette dernière est capturée comme aux échecs. Mais c'est la différence de prise qui apporte un interêt nouveau à ce jeu. Aux échecs, quand vous prenez une pièce adverse, elle quitte définitivement l'échiquier pour tout le restant de la partie. Au shogi, vous garder cette pièce en réserve et plus tard dans la partie, vous pourrez la parachuter sur toute case libre (moyennant quelques restrictions). Le parachutage remplace le déplacement d'une pièce d'une case à une autre. Une pièce ne peut être ... parachutée que non promue, même si elle avait été promue avant sa capture. Quand vous parachutez une pièce prise sur l'échiquier, elle vous appartient. La prise d'une pièce ennemie a donc un double intérêt. Un des aspects du parachutage est que vous pouvez tout à fait sacrifier une pièce de valeur dans une case contre une pièce de valeur inférieure, simplement parce que vous voulez être capable de parachuter cette dernière sur une autre case dans les quelques coups à venir.

4. COMME AUX ECHECS, IL FAUT METTRE LE ROI MAT

Quand un roi est attaqué, il est dit "en échec", tout comme aux échecs, et le joueur en échec doit s'en soustraire immédiatement, en déplacant son propre roi, en capturant la pièce qui provoque cet échec ou en interposant une pièce entre les deux. Si un roi est attaqué et qu'il n'y a aucun moyen de le sauver, le joueur a été mis en échec et mat. Les pièces ètant toutes en jeu pendant la partie, il est très rare qu'une partie de shogi se termine sur un nul. Aux échecs le nombre de pièces sur l'échiquier se reduit durant la partie et la partie risque de se terminer plus souvent par une partie nulle. Ceux qui trouvent ennuyeuses les parties de maîtres en échecs parce qu'elles sont trop souvent nulles (55 % au plus), n'ont rien à craindre au shogi.

4.1. Comment programmer le shogi ?

On peut appliquer au shogi la plupart des principes de la programmation du jeu d'échecs. On commence par développer et explorer un arbre de jeu, le problème essentiel étant l'énorme facteur de croissance causé par le grand nombre de pièces (40 au lieu d'un maximum de 32) et la possibilité d'un parachutage. Si vous n'avez "en reserve" qu'un seul type de pièces capturées, il y aura 42 cases ou plus sur lesquelles on pourra la parachuter. On voit que le nombre de coups légaux disponibles peut facilement aller jusqu'à 150 ou 200, dès que deux ou trois pièces ennemies ont été prises. Il est clair qu'il faut trouver un moyen quelconque d'en réduire le nombre et produire une liste maniable de coups plausibles. On doit dans ce cas utiliser des raisonnements heuristiques intelligents propres au Shogi, connus par les adeptes sous le nom de "proverbes".

Ceux qui sont intéressés par l'écriture d'un programme d'échecs n'ont qu'à se reporter à l'énorme littérature existante pour découvrir des raisonnements heuristiques qui peuvent être employés dans un générateur de coups plausibles ou dans un mécanisme d'évaluation. On a aussi beaucoup écrit sur le Shogi, mais par malchance, presque tout est publié en japonais, et si votre japonais est aussi mauvais que le mien, vous hésiterez à faire un safari dans des volumes remplis de symboles mystérieux. Je manque de symboles mystérieux. Je manque de place pour traiter ces divers raisonnements heuristiques. D'ailleurs, je vous recommande de jeter un coup d'oeil au dos du livre de Fairbain ; il existe une nombreuse littérature à ce sujet.

Par ailleurs, ceux qui désirent rendre leur programme le plus fort possible devraient s'inscrire à l'association de Shogi et essayer d'obtenir les numéros du magazine "Shogi" où les principaux proverbes sont expliqués. Dès que vous avez compris un proverbe, c'est chose facile que de le convertir sous forme numérique de façon à l'intégrer au mécanisme d'évaluation de la plausibilité.

4.2. Deux types d'ouverture sont possibles.

L'ordre exact dans lequel les coups d'ouverture sont joués n'est pas aussi problématique au Shogi qu'aux échecs. Le plus important dans l'ouverture au Shogi réside dans les cases sur lèsquelles on met ses pièces et non dans l'ordre exact qui les y amène. Puisqu'il n'est pas nécessaire que votre programme de Shogi ait accès à d'importantes tables de variantes d'ouvertures, il vous suffit de préparer une méthode qui incite . le programme à jouer des coups qui amèneront les pièces sur les bonnes cases. Une méthode simple consiste à examiner chacune des pièces d'une formation désirée et à déterminer, à un moment donné de combien de coups chacune d'entre elles est distante de sa case d'arrivée et non dans l'ordre exact qui les y amène ..

Table des matières

accepté ?

\\INT:RE:2.
\\PO:RE:11.
\\IMP:L:T.
\\PA:5.
\\TA:4.
\DO.
\TI.
8.3.1."BON" EXEMPLE
\AU.
Mathieu - Schmitz \ David Levy
\DA.
Lausanne, 12 - avril - 1983
\RE.Le texte source se compose de caract\'eres constituant les commandes du niveau global, les commandes du niveau local et le texte.
\\CR.Le contenu du texte n'a aucune importance. Les commandes du niveau global sont les commandes situ\'ees avant la s\'equence <\DOCU.>. Elles ont \'et\'e choisies afin de pr\'esenter les diff\'erentes commandes de formatage et afin de pr\'esenter un texte "propre".
\\CR.Des commandes de niveau global et des commandes de niveau local seront pr\'esentes dans ce texte. Elles seront pr\'esent\'ees en majuscules et sous leur forme r\'eduite, ceci n'\'etant en rien une contrainte.
\CH::1.\EN.INTRODUCTION
\PA.Je vous pr\'esente aujourd'hui un jeu que vous n'avez probablement
\\FO : GR.jamais\\FIN : FO. rencontr\'e.
\\CR.Cousin des échecs, sa complexité est plus grande d'o\'u des parties passionnantes qui se poursuivent sur un nombre de coups nettememt sup\'erieur \'a celui des \'echecs. Ce jeu est si populaire au Japon que les meilleurs deviennent souvent millionnaires et sont plus v\'en\'er\'es que Bjorn Borg en Su\'ede ou Keeving Keegan en Angleterre. Je conseille \'a tous ceux qui aiment les \'echecs "occidentaux" de t\^ater du shogi, les \'echecs japonais.
\CH::1.\EN.INITIATION
\PA.Ceux de mes amis qui jouent au shogi le trouvent passionnant et d'autres apprecieraient le jeu apr\'es une heure ou deux d'apprentissage. Si le Japon est la "Mecque" du shogi, une organisation s'est d\'evelopp\'ee dans le monde occidental durant ces derni\'eres ann\'ees ; son but : populariser le jeu hors de son pays natal. T\^ache qui s'avere difficile \'etant donn\'e la non popularit\'e de cette discipline et la concurrence que lui fait le traditionnel jeu d'échecs qui lui est r\'epandu dans le monde entier et se popularise grace \'a des concours internationaux et \'a l'affrontement des g\'eants de ce jeu. L'association de shogi accueille les nouveaux membres et fournit des jeux de shogi et des livres d'initiation \'a ceux qui ne peuvent les trouver ailleurs. Elle publie \'egalement un magazine et propose des rencontres \'a Londres.
\PA.Gr\^ace \'a ces efforts, des tournois de shogi ont eu lieu un peu partout dans le monde. Mon coll\'egue americain Larry Kauffman, ma\^itre international d'\'echecs, est devenu compl\'etement fou de ce jeu; il s'est m\^eme rendu au Japon dans l'espoir de devenir un grand champion de cette discipline.
\CH::2.\EN.\\FF.Comment jouer au shogi?
\PA.Chaque joueur entame la partie avec 20 pi\'eces en bois ou en plastique. Ces pi\'eces sont de la m\^eme couleur. Le jeu se d\'eroule sur un \'echiquier 9 * 9 (connait-on un microprocesseur \'a 9 bits) avec deux arm\'ees situ\'ees dans la position suivante
\FIG.25,70.la place est r\'eserv\'ee sur la page pour ins\'erer une figure 4 de nom sp\'ecifi\'e par l'utilisateur ; ceci constitue la l\'egende, une r\'ef\'erence est ins\'er\'ee dans le texte \'a l'endroit de son "appel"

\PA.La meilleure facon d'apprendre est d'acheter le livre de FAIRBANK avec un guide pr\'es de vous.
\NO.ce livre s'intitule : "How to play Shogi"\FIN:NOTE. Les jeux se composent de pi\'eces marqu\'ees avec des caract\'eres japonais mais sont import\'ees avec les pi\'eces marqu\'ees uniquement de caract\'eres occidentaux et des fl\'eches qui vous indiquent leur sens de d\'eplacement. L'apprentissage du shogi n'est pas plus long que celui des \'echecs, et je me suis laiss\'e dire de source s\^ure que l'on s'accoutume aux symboles japonais bien plus vite que l'on ne le penserait. Le but de cet article \'etant de permettre au lecteur d'\'ecrire son propre programme de shogi, je commencerai donc par r\'esumer les r\'egles et les coups du jeu.
\CH::2.\EN.Les pi\'eces et leur d\'eplacement
\PA.\\SO:C.Le roi\\FIN:S. : chaque joueur a un roi et comme aux \'echecs, le but du jeu est de mettre \'echec et mat le roi adverse. Comme aux \'echecs, le roi peut se d\'eplacer d'une case dans toutes les directions (horizontale, verticale ou diagonale).
\PA.\\SO:C.Le g\'en\'eral d'or\\FIN:SO. : en d\'ebut de partie chaque joueur dispose de deux de ces pi\'eces qui avancent d'une case \'a la fois \'a la verticale, \'a l'horizontale et en diagonale vers l'avant. Ils ne peuvent pas se d\'eplacer en diagonale vers l'arri\'ere.
\PA.\\SO:C.Le g\'en\'eral d'argent\\FIN:SO. : chaque joueur dispose \'egalement de deux de ces pi\'eces. Celles-ci avancent d'une case en diagonale ou vers l'avant. Elles ne peuvent ni aller sur le cot\'e ni reculer directement.
\PA.\\SO :C.Le cavalier\\FIN :SO. : chaque joueur \'a deux cavaliers dont le d\'eplacement est
semblable \'a celui des \'echecs mais avec la restriction suivante : ils ne peuvent avancer que de deux cases puis aller sur la case \'a gauche ou \'a droite, alors qu'aux \'echecs le cavalier a huit coups \'a sa disposition depuis une case centrale sur un \'echiquier vide, un cavalier de shogi n'aura que deux possibilit\'es, mais il peut sauter comme aux \'echecs.
\PA.\\SO:C.Le lancier\\FIN:SO. : chacun des deux lanciers avance droit devant lui autant qu'il le desire, mais ils ne peuvent en aucun cas sauter une pi\'ece de l'adversaire.\PA.
\\SO:C.La tour\\FIN:SO. : la tour de shogi se d\'eplace exactement comme son homologue des \'echecs c'est-\'a-dire en ligne droite, d'autant de cases qu'elle le d\'esire. Comme il n'y a pas de dame au shogi, on consid\'ere g\'en\'eralement que la tour est la pi\'ece la plus puissante de toutes celles de l'\'echiquier.
\PA.\\SO:C.Le fou\\FIN:SO. : cette pi\'ece se d\'eplace \'egalement comme un fou dans le jeu d'\'echecs, de n'importe quel nombre de cases en diagonale.
\PA.\\SO:C.Le pion\\FIN:SO. : comme aux \'echecs, la position initiale de shogi comporte une rang\'ee de pions qui barre la largeur de l'\'echiquier. Chaque joueur commence la partie avec neuf pions qui se d\'eplacent vers l'avant. Il n'y a pas de d\'eplacement double lors du premier coup ; il n'y a pas de prise en diagonale et il n'y a pas de prise en passant. \\CR.Pour r\'esumer voici une liste des pi\'eces :
\LISTE.le pion
\\\le cavalier
\\\la tour
\\\le lancier
\\\le g\'en\'eral d'or
\\\le g\'en\'eral d'argent
\\\le roi
\CH::2.\EN.Attention aux pi\'eces promues
\PA.Tandis qu'aux \'echecs seuls les pions peuvent \^etre promus et deviennent des pi\'eces de valeur sup\'erieure, l'inter\^et du shogi provient de ce que certaines autres pi\'eces peuvent

\'egalement \^etre promues. On r\'ealise un coup de promotion en d\'eplacant en partie ou compl\'etement dans la zone de promotion de l'adversaire (les trois derni\'eres rang\'ees les plus \'eloign\'ees de vous). La promotion a lieu \'a la fin d'un tel coup mais il n'est pas toujours obligatoire de promouvoir au shogi. Plusieures pi\'eces peuvent \^etre promues.

\CH::3.\EN.Cas du g\'en\'eral d'argent

\PA.Une fois promu il se d\'eplace comme un g\'en\'eral d'or. Sur votre jeu de shogi, le g\'en\'eral d'argent peut \^etre retourn\'e et sur l'autre cot\'e, vous verrez le symbole d'un g\'en\'eral d'argent qui vient d'\^etre promu.

\CH::4.\EN.Cas des autres pi\'eces

\PA.Le cavalier, le lancier, le pion se d\'eplacent \'egalement comme le g\'en\'eral d'or une fois qu'ils viennent d'etre promus.

\\CR.La tour lorsqu'elle est promue garde sa capacit\'e originelle de d'\'eplacement d'un nombre quelconque de cases \'a l'horizontale ou \'a la verticale mais elle acquiert la possibilit\'e de se d\'eplacer d'une case en diagonale. Le fou, de la m\^eme mani\'ere acquiert la possibilit\'e de se d\'eplacer d'une case supl\'ementaire \'a l'horizontale ou \'a la verticale. Si un pion ou un lancier se d\'eplace jusqu'au dernier rang ou si un cavalier atteint un des deux premiers rangs, la promotion est obligatoire. Dans tous les autres cas la promotion est optionnelle.

Il vaut mieux en g\'en\'eral promouvoir ; cela assure un avantage certain sur l'adversaire \'a condition qu'il n'ait d\'ej\'a pas eu l'occasion de promouvoir

\CH::1.\EN.UNE PIECE PRISE AU JEU PEUT ETRE REMISE EN JEU

\PA.Si un joueur d\'eplace une de ses pi\'eces sur une case occup\'eee par l'adversaire, cette derni\'ere est captur\'ee comme aux \'echecs. Mais c'est la diff\'erence de prise qui apporte un inter\^et nouveau \'a ce jeu. Aux \'echecs, quand vous prenez une pi\'ece adverse, elle quitte d\'efinitivement l'\'echiquier pour tout le restant de la partie. Au shogi, vous garder cette pi\'ece en r\'eserve et plus tard dans la partie, vous pourrez la parachuter sur toute case libre (moyennant quelques restrictions). Le parachutage remplace le d\'eplacement d'une pi\'ece d'une case \'a une autre. Une pi\'ece ne peut etre que parachut\'ee que non promue, m\^eme si elle avait \'et\'e promue avant sa capture. Quand vous parachutez une pi\'ece prise sur l'\'echiquier, elle vous appartient. La prise d'une pi\'ece ennemie a donc un double int\'er\^et. Un des aspects du parachutage est que vous pouvez tout \'a fait sacrifier une pi\'ece de valeur dans une case contre une pi\'ece de valeur inf\'erieure, simplement parce que vous voulez \^etre capable de parachuter cette deni\'ere sur une autre case dans les quelques coups \'a venir .

\CH::1.\EN.COMME AUX ECHECS, IL FAUT METTRE LE ROI MAT

\PA.Quand un roi est attaqu\'e, il est dit "en \'echec", tout comme aux \'echecs, et le joueur en \'echec doit s'en soustraire imm\'ediatement, en d\'eplacant son propre roi, en capturant la pi\'ece qui provoque cet \'echec ou en interposant une pi\'ece entre les deux. Si un roi est attaqu\'e et qu'il n'y a aucun moyen de le sauver, le joueur a \'et\'e mis en \'echec et mat. Les pi\'eces \'etant toutes en jeu pendant la partie, il est tr\'es rare qu'une partie de shogi se termine sur un nul. Aux \'echecs le nombre de pi\'eces sur l'\'echiquier se reduit durant la partie et la partie risque de se terminer plus souvent par une partie nulle. Ceux qui trouvent ennuyeuses les parties de ma\^itres en \'echecs parce qu'elles sont trop souvent nulles (55 % au plus), n'ont rien \'a craindre au shogi.

\CH::2.

\EN.

Comment programmer le shogi ?

\PA.On peut appliquer au shogi la plupart des principes de la programmation du jeu d'\'echecs. On commence par d\'evelopper et explorer un arbre de jeu, le probl\'eme essentiel \'etant l'\'enorme facteur de croissance caus\'e par

le grand nombre de pi\'eces (40 au lieu d'un maximum de 32) et la possibilit\'e d'un parachutage. Si vous n'avez "en reserve" aucun seul type de pi\'eces captur\'ees, il y aura 42 cases ou plus sur lesquelles on pourra la parachuter. On voit que le nombre de coups l\'egaux disponibles peut facilement aller jusqu'\'a 150 ou 200, d\'es que deux ou trois pi\'eces ennemies ont \'et\'e prises. Il est clair qu'il faut trouver un moyen quelconque d'en r\'eduire le nombre et produire une liste maniable de coups plausibles. On doit dans ce cas utiliser des raisonnements heuristiques intelligents propres au Shogi, connus par les adeptes sous le nom de "proverbes".

\PA.Ceux qui sont int\'eress\'es par l'\'ecriture d'un programme d'\'echecs n'ont qu'\'a se reporter \'a l'\'enorme litt\'erature existante pour d\'ecouvrir des raisonnements heuristiques qui peuvent \^etre employ\'es dans un g\'en\'erateur de coups plausibles ou dans un m\'ecanisme d'\'evaluation. On a aussi
beaucoup \'ecrit sur le Shogi, mais par malchance, presque tout est publi\'e en japonais,
et si votre japonais est aussi mauvais que le mien,
vous h\'esiterez \'a faire un safari dans des volumes remplis de symboles myst\'erieux. Je manque de symboles myst\'erieux. Je manque de place pour traiter ces divers raisonnements heuristiques. D'ailleurs, je vous recommande de jeter un coup d'oeil au dos du livre de Fairbain ; il existe une nombreuse litt\'erature \'a ce sujet.

\PA.Par ailleurs,ceux qui d\'esirent rendre leur programme le plus fort possible devraient s'inscrire \'a l'association de Shogi et essayer d'obtenir les num\'eros du magazine "Shogi" o\'u les principaux proverbes sont expliqu\'es. D\'es que vous avez compris un proverbe, c'est chose facile que de le convertir sous forme num\'erique de fa\,con \'a l'int\'egrer au m\'ecanisme d'\'evaluation de la plausibilit\'e.

\CH:2.\EN.Deux types d'ouverture sont possibles.

\PA.L'ordre exact dans lequel les coups d'ouverture sont jou\'es n'est pas aussi probl\'ematiques au Shogi qu'aux \'echecs. Le plus important dans l'
ouverture au Shogi r\'eside dans les cases sur lesquelles on met ses pi\'eces et non dans l'ordre exact qui les y am\'ene .. Puisqu'il n'est pas n\'ecessaire que votre programme de Shogi ait acc\'es \'a d'importantes tables de variantes d'ouvertures, il vous suffit de pr\'eparer une m\'ethode qui incite le programme \'a jouer des coups qui am\'eneront les pi\'eces sur les bonnes cases. Une m\'ethode simple consiste \'a examiner chacune des pi\'eces d'une formation d\'esir\'ee et \'a d\'eterminer, \'a un moment donn\'e de combien de coups chacune d'entre elles est distante de sa case d'arriv\'ee et non dans l'ordre exact qui les y am\'ene ..

8.3.2. "MAUVAIS" EXEMPLE

Mathieu - Schmitz & David Levy

Lausanne, 12 - avril - 1983

Le texte source se compose de caractères constituant les commandes du niveau global, les commandes du niveau local et le texte.
Le contenu du texte n'a aucune importance. Les commandes du niveau global sont les commandes situées avant la séquence <\DOCU.>. Elles ont été choisies afin de présenter les différentes commandes de formatage ; elles n'ont pas été choisies pour obtenir une présentation convenable du texte .
La liste des commandes n'est pas exhaustive ; nous trouverons des commandes de niveau global et des commandes de niveau local.
Parmi les commandes du niveau global, nous trouverons, par exemple, un numérota ge de pages du document - numérotage en haut de page, une table des matières à trois niveaux , une police numéro 14 pour les notes bas de page, un interligne de 2 millimètres pour l' élément de type résumé, les éléments d'identifications du document sur une première page, centrés et non numérotés...
Parmi les commandes de niveau local - modifiant le formatage de l'élément logique courant, nous trouverons, par exemple, la spécification d'une fonte, d'une police, et d'un soulignement différent de celui défini soit au niveau global, soit prédéfini, l'introduction de commandes de saut de page, de saut de ligne, de modification des marges . . .
Les commandes des deux niveaux sont EN MAJUSCULES ET SOUS LEUR FORME REDUITE ; *aucune de ces conventions n'est une contrainte, mais elles facilitent la visualisation.*

1. INTRODUCTION

Je vous présente aujourd'hui un jeu que vous n'avez probablement **jamais** rencontré. Cousin des echecs, sa complexité est plus grande d'où des parties passionnantes qui se poursuivent sur un nombre de coups nettement supérieur à celui des échecs. Ce jeu est si populaire au Japon que les meilleurs deviennent souvent millionnaires et sont plus vénérés que Bjorn Borg en Suède ou Keeving Keegan en Angleterre. Je conseille à tous ceux qui aiment les échecs "occidentaux" de tâter du shogi, les échecs japonais.

2. INITIATION

Ceux de mes amis qui jouent au shogi le trouve passionnant et d'autres apprécieraient le jeu après une heure ou deux d'apprentissage. Si le Japon est la "Mecque" du shogi, une organisation s'est développée dans le monde occidental durant ces dernières années ; son but : populariser le jeu hors de son pays natal. Tâche qui s'avére difficile étant donné la non-popularité de cette discipline et la concurrence que lui fait le traditionnel jeu d'echec qui, lui, est répandu dans le monde entier et se popularise grâce à des concours internationaux et à l'affrontement des géants de ce jeu. L'association de shogi accueille les nouveaux membres et fournit des jeux de shogi et des livres d'initiation à ceux qui ne peuvent les trouver ailleurs. Elle publie également un magazine et propose des rencontres à Londres.

Grâce à ces efforts, des tournois de shogi ont eu lieu un peu partout dans le monde. Mon collègue américain Larry Kauffman, maître international d'échecs, est devenu complètement fou de ce jeu; il s'est même rendu au Japon dans l'espoir de devenir un grand champion de cette discipline.

2.1. Comment jouer au shogi?

Chaque joueur entame la partie avec 20 pièces en bois ou en plastique. Ces pièces sont de la même couleur. Le jeu se déroule sur un échiquier 9 * 9 (connaît-on un microprocesseur à 9 bits) avec deux armées situées dans la position suivante (fig. 1)

- Fig. 1 - la place est réservée sur la page pour insérer une figure de nom spécifié par l'utilisateur ; ceci constitue la légende, une référence est insérée dans le texte à l'endroit de son "appel"

La meilleure facon d'apprendre est d'acheter le livre de FAIRBANK avec un guide près de vous. (nb. 1) Les jeux se composent de pièces marquées avec des caractères japonais mais sont importés avec les pièces marquées uniquement de caractères occidentaux et des flèches qui vous indiquent leur sens de déplacement. L'apprentissage du shogi n'est pas plus long que celui des échecs, et je me suis laissé dire de source sûre que l'on s'accoutume aux symboles japonais bien plus vite que l'on ne le penserait. Le but de cet article étant de permettre au lecteur d'écrire son propre programme de shogi, je commencerai donc par résumer les règles et les coups du jeu.

NB: 1 ce livre s'intitule : "How to play Shogi"

2.2. Les pièces et leur déplacement

Le roi : chaque joueur a un roi et comme aux échecs, le but du jeu est de mettre échec et mat le roi adverse. Comme aux échecs, le roi peut se déplacer d'une case dans toutes les directions (horizontale, verticale ou diagonale).

Le général d'or : en début de partie chaque joueur dispose de deux de ces pièces qui avancent d'une case à la fois à la verticale, à l'horizontale et en diagonale vers l'avant. Ils ne peuvent pas se déplacer en diagonale vers l'arrière.

Le général d'argent : chaque joueur dispose également de deux de ces pièces. Celles-ci avancent d'une case en diagonale ou vers l'avant. Elles ne peuvent ni aller sur le côté ni reculer directement.

Le cavalier : chaque joueur a deux cavaliers dont le déplacement est semblable à celui des échecs mais avec la restriction suivante : ils ne peuvent avancer que de deux cases puis aller sur la case à gauche ou à droite, alors qu'aux échecs le cavalier a huit coups à sa disposition depuis une case centrale sur un échiquier vide, un cavalier de shogi n'aura que deux possibilités, mais il peut sauter comme aux échecs.

Le lancier : chacun des deux lanciers avance droit devant lui autant qu'il le desire, mais ils ne peuvent en aucun cas sauter une pièce de l'adversaire.

La tour : la tour de shogi se déplace exactement comme son homologue des échecs c'est-à-dire en ligne droite, d'autant de cases qu'elle le désire. Comme il n'y a pas de dame au shogi, on considère généralement que la tour est la pièce la plus puissante de toutes celles de l'échiquier.

Le fou: cette pièce se déplace également comme un fou dans le jeu d'échec de n'importe quel nombre de cases en diagonale.

Le pion : comme aux échecs, la position initiale de shogi comporte une rangée de pions qui barre la largeur de l'échiquier. Chaque joueur commence la partie avec neuf pions qui se déplacent vers l'avant. Il n'y a pas de déplacement double lors du premier coup ; il n'y a pas de prise en diagonale et il n'y a pas de prise en passant.

Pour résumer voici une liste des pièces :

- le pion
- le cavalier
- la tour
- le lancier
- le général d'or
- le général d'argent
- le roi

2.3. Attention aux pièces promues

Tandis qu'aux échecs seuls les pions peuvent être promus et deviennent des pièces de valeur supérieure, l'interêt du shogi provient de ce que certaines autres pièces peuvent également être promues. On réalise un coup de promotion en déplacant en partie ou complètement dans la zone de promotion de l'adversaire (les trois dernières rangées les plus éloignées de vous). La promotion a lieu à la fin d'un tel coup mais il n'est pas toujours obligatoire de promouvoir au shogi. Plusieurs pièces peuvent être promues.

2.3.1. Cas du général d'argent

Une fois promu, il se déplace comme un général d'or. Sur votre jeu de shogi, le général d'argent peut être retourné et sur l'autre coté, vous verrez le symbole d'un général d'argent qui vient d'être promu.

2.3.1.1. Cas des autres pièces

Le cavalier, le lancier, le pion se déplacent également comme le général d'or une fois qu'ils viennent d'être promus.
La tour lorsqu'elle est promue, (nb. 2) d'un nombre quelconque de cases à l'horizontale ou à la verticale mais elle acquiert la possibilité de se déplacer d'une case en diagonale. Le fou, de la même manière acquiert la possibilité de se déplacer d'une case suplémentaire à l'horizontale ou à la verticale. Si un pion ou un lancier se déplace jusqu'au dernier rang ou si un cavalier atteint un des deux premiers rangs, la promotion est obligatoire. Dans tous les autres cas la promotion est optionnelle. (nb. 3)

3. UNE PIECE PRISE AU JEU PEUT ETRE REMISE EN JEU

Si un joueur déplace une de ses pièces sur une case occupée par l'adversaire, cette dernière est capturée comme aux échecs. Mais c'est la différence de prise qui apporte un interêt nouveau à ce jeu. Aux échecs, quand vous prenez une pièce adverse, elle quitte définitivement l'échiquier pour tout le restant de la partie. Au shogi, vous garder cette pièce en réserve et plus tard dans la partie, vous pourrez la parachuter sur toute case libre (moyennant quelques restrictions). Le parachutage remplace le déplacement d'une pièce d'une case à une autre. Une pièce ne peut être parachutée que non promue, même si elle avait été promue avant sa capture. Quand vous parachutez une pièce prise sur l'échiquier, elle vous appartient. La prise d'une pièce ennemie a donc un double intérêt. Un des aspect du parachutage est que vous pouvez tout à fait sacrifier une pièce de valeur dans une case contre une pièce de valeur inférieure, simplement parce que vous voulez être capable de parachuter cette dernière sur une autre case dans les quelques coups à venir.

4. COMME AUX ECHECS, IL FAUT METTRE LE ROI MAT

NB: 2 garde sa capacité originelle de déplacement

NB: 3 il vaut mieux en général promouvoir ; cela assure un avantage certain sur l'adversaire à condition qu'il n'ait déjà pas eu l'occasion de promouvoir

Quand un roi est attaqué, il est dit "en échec", tout comme aux échecs, et le joueur en échec doit s'en soustraire immédiatement, en déplacant son propre roi, en capturant la pièce qui provoque cet échec ou en interposant une pièce entre les deux. Si un roi est attaqué et qu'il n'y a aucun moyen de le sauver, le joueur a été mis en échec et mat. Les pièces ètant toutes en jeu pendant la partie, il est très rare qu'une partie de shogi se termine sur un nul. Aux échecs le nombre de pièces sur l'échiquier se reduit durant la partie et la partie risque de se terminer plus souvent par une partie nulle. Ceux qui trouvent ennuyeuses les parties de maîtres en échecs parce qu'elles sont trop souvent nulles (55 % au plus), n'ont rien à craindre au shogi.

4.1. Comment programmer le shogi?

On peut appliquer au shogi la plupart des principes de la programmation du jeu d'échecs. On commence par développer et explorer un arbre de jeu, le problème essentiel étant l'énorme facteur de croissance causé par le grand nombre de pièces (40 au lieu d'un maximum de 32) et la possibilité d'un parachutage. Si vous n'avez "en reserve" qu'un seul type de pièces capturées, il y aura 42 cases ou plus sur lesquelles on pourra la parachuter. On voit que le nombre de coups légaux disponibles peut facilement aller jusqu'à 150 ou 200, dès que deux ou trois pièces ennemies ont été prises. Il est clair qu'il faut trouver un moyen quelconque d'en réduire le nombre et produire une liste maniable de coups plausibles. On doit dans ce cas utiliser des raisonnements heuristiques intelligents propres au Shogi, connus par les adeptes sous le nom de "proverbes".

Ceux qui sont intéressés par l'écriture d'un programme d'échecs n'ont qu'à se reporter à l'énorme littérature existante pour découvrir des raisonnements heuristiques qui peuvent être employés dans un générateur de coups plausibles ou dans un mécanisme d'évaluation. On a aussi beaucoup écrit sur le Shogi, mais par malchance, presque tout est publié en japonais, et si votre japonais est aussi mauvais que le mien, vous hésiterez à faire un safari dans des volumes remplis de symboles mystérieux. Je manque de symboles mystérieux. Je manque de place pour traiter ces divers raisonnements heuristiques. D'ailleurs, je vous recommande de jeter un coup d'oeil au dos du livre de Fairbain ; il existe une nombreuse littérature à ce sujet.

Par ailleurs, ceux qui désirent rendre leur programme le plus fort possible devraient s'inscrire à l'association de Shogi et essayer d'obtenir les numéros du magazine "Shogi" où les principaux proverbes sont expliqués. Dès que vous avez compris un proverbe, c'est chose facile que de le convertir sous forme numérique de façon à l'intégrer au mécanisme d'évaluation de la plausibilité.

4.2. Deux types d'ouverture sont possibles.

L'ordre exact dans lequel les coups d'ouverture sont joués n'est pas aussi problématique au Shogi qu'aux échecs. L plus important dans l'ouverture au Shogi réside dans les cases sur lesquelles on met ses pièces et non dans l'ordre exact qui les y amène. Puisqu'il n'est pas nécessaire que **votre programme de Shogi ait accès à d'importantes tables** de variantes d'ouvertures, il vous suffit de préparer une méthode qui incite le programme à jouer des coups qui amèneront les pièces sur les bonnes cases. Une méthode simple consiste à examiner chacune des pièces d'une formation désirée et à déterminer, à un moment donné de combien de coups chacune d'entre elles est distante de sa case d'arrivée et non dans l'ordre exact qui les y amène .

Table des matières

```
\\DE:P:10
\\INT:RE: 2.
\\FO:RE:ITA.
\\PO:RE:11.
\\PO:NO:14.
\\SO:EN:N:3: C.
\\PA:S.
\\TA:3.
\\UN.
\DO.
\TI.8.3.2. "MAUVAIS" EXEMPLE
\AU.Mathieu - Schmitz & David Levy
\DA.Lausanne, 12 - avril - 1983
\RE.Le texte source se compose de caract\'eres constituant les commandes
  du niveau global, les commandes du niveau local et le texte.
\\CR.Le contenu du texte n'a aucune
  importance. Les commandes du niveau global sont les
  commandes situ\'ees avant la s\'equence\\SOU : C. <\DOCU.>.\\FIN:S. Elles
  ont \'et\'e choisies afin de pr\'esenter les diff\'erentes commandes
  de formatage ; elles n'ont pas \'et\'e choisies pour obtenir une
  pr\'esentation convenable du texte .
\\CR.La liste des commandes n'est pas exhaustive ; nous trouverons des
  commandes de niveau globale et des commandes de niveau local.
\\CR.Parmi les
\\SOU : DI.commandes du niveau global\\FIN : SO., nous trouverons,
par exemple, un num\'erota ge de pages du document
  - num\'erotage en haut de page, une table des mati\'eres \'a
  trois niveaux,
  , une police num\'ero 14 pour les notes bas de page,
  un interligne de 2 millim\'etres pour l' \'el\'ement de type
  r\'esum\'e, les \'el\'ements d'identifications du document sur
  une premi\'ere page, centr\'es et non num\'erote\'es.
\\CR.Parmi les \\SOU : DI.commandes de niveau local\\FIN : SO. -
modifiant le formatage de l'\'el\'ement logique courant,
  nous trouverons, par exemple, la sp\'ecification d'une fonte, d'une police,
  et d'un soulignement diff\'erent de celui d\'efini soit au niveau global, soit
  pr\'ede\'efini, l'introduction de commandes de saut de page, de saut de ligne,
    de modification des marges . . .
\\CR.Les commandes des deux niveaux sont
  \\POL : 14.en majuscule et sous leur forme r\'eduite\\FIN : POL. ;
  aucune de ces conventions n'est une contrainte, mais elles facilitent
  la visualisation.
\CH::1.
\EN.INTRODUCTION
\PA.Je vous pr\'esente aujourd'hui un jeu que vous n'avez probablement
\\FO : GR.jamais
\\FIN : FO. rencontr\'e.
\\CR.Cousin des echecs, sa complexite est plus grande d'o\'u des parties
  passionnantes qui se poursuivent sur un nombre de coups nettememt sup\'erieur
  \'a celui des \'echecs. Ce jeu est si populaire au Japon que les meilleurs
  deviennent souvent millionnaires et sont plus v\'en\'er\'es que Bjorn Borg
  en Su\'ede ou Keeving Keegan en Angleterre. Je conseille \'a tous
  ceux qui aiment les \'echecs "occidentaux" de t\^ater du shogi, les \'echecs
  japonais.
\CH::1.
\EN.\\PO : 10.\\FO : ITAL.INITIATION
\PA.Ceux de mes amis qui jouent au shogi le trouve" passionnant et
  d'autres  appreciraient le jeu apr\'es une heure ou deux d'apprentissage.
```

Si le Japon est la "Mecque" du shogi, une organisation s'est d\'evelopp\'ee dans le monde occidental durant ces derni\'eres ann\'ees ; son but : populariser le jeu hors de son pays natal. T\^ache qui s'avère difficile étant donn\'e la non popularit\'e de cette discipline et la concurrence que lui fait le traditionnel jeu d'echec qui lui est r\'epandu dans le monde entier et se popularise grâce \'a des concours internationaux et \'a l' affrontement des g\'eants de ce jeu. L'association de shogi acceuille les nouveaux membres et fournit des jeux de shogi et des livres d'initiation \'a ceux qui ne peuvent les trouver ailleurs. Elle publie \'egalement un magazine et propose des rencontres \'a Londres.

\PA.\\PA. Gr\^ace \'a ces efforts, des tournois de shogi ont eu lieu un peu partout dans le monde. Mon coll\'egue americain Larry Kauffman, ma\^itre international d'\'echecs, est devenu compl\'etement fou de ce jeu; il s'est m\^eme rendu au Japon dans l'espoir de devenir un grand champion de cette discipline.

\CH::2.

\EN.Comment jouer au shogi?\PA. Chaque joueur entame la partie avec 20 pi\'eces en bois ou en plastique. Ces pi\'eces sont de la m\^eme couleur. Le jeu se d\'eroule sur un \'echiquier 9 * 9 (connait-on un microprocesseur \'a 9 bits) avec deux arm\'ees situ\'ees dans la position suivante

\FIG:25,70.la place est r\'eserv\'ee sur la page pour ins\'erer une figure de nom sp\'ecifi\'e par l'utilisateur ; ceci constitue la l\'egende, une r\'ef\'erence est ins\'er\'ee dans le texte \'a l'endroit de son "appel"

\PA. La meilleure facon d'apprendre est d'acheter le livre de FAIRBANK avec un guide pr\'es de vous.

\NO. ce livre s'intitule : "How to play Shogi"

\FIN:NOTE. Les jeux se composent de pi\'eces marqu\'ees avec des caract\'eres japonais mais sont import\'ees avec les pi\'eces marqu\'ees uniquement de caract\'eres occidentaux et des fl\'eches qui vous indiquent leur sens de d\'eplacement. L'apprentissage du shogi n'est pas plus long que celui des \'echecs, et je me suis laiss\'e dire de source s\^ure que l'on s'accoutume aux symboles japonais bien plus vite que l'on ne le penserait. Le but de cet article \'etant de permettre au lecteur d'\'ecrire son propre programme de shogi, je commencerai donc par r\'esumer les r\'egles et les coups du jeu.

\CH::2.

\EN.Les pi\'eces et leur d\'eplacement

\PA.\\DE:20.Le roi : chaque joueur a un roi et comme aux \'echecs, le but du jeu est de mettre \'echec et mat le roi adverse. Comme aux \'echecs, le roi peut se d\'eplacer d'une case dans toutes les directions (horizontale, verticale ou diagonale).

\PA.\\SO:C. Le g\'en\'eral d'or
\\FIN:SO. : en d\'ebut de partie chaque joueur dispose de deux de ces pi\'eces qui avancent d'une case \'a la fois \'a la verticale, \'a l'horizontale et en diagonale vers l'avant. Ils ne peuvent pas se d\'eplacer en diagonale vers l'arri\'ere.

\PA.\\SO:C.Le g\'en\'eral d'argent\\FIN:SO. : chaque joueur dispose \'egalement de deux de ces pi\'eces. Celles-ci avancent d'une case en diagonale ou vers l'avant. Elles ne peuvent ni aller sur le cot\'e ni reculer directement.

\PA.Le cavalier : chaque joueur \'a deux cavaliers dont le d\'eplacement est semblable \'a celui des \'echecs mais avec la restriction suivante : ils ne peuvent avancer que de deux cases puis aller sur la case \'a gauche ou \'a droite, alors qu'aux \'echecs le cavalier a huit coups \'a sa disposition depuis une case centrale sur un \'echiquier vide, un cavalier de shogi

n'aura que deux possibilit\'es, mais il peut sauter comme aux \'echecs.
\PA.\\SO:D.Le lancier\\FIN:SO. : chacun des deux
lanciers avance droit devant lui autant qu'il
le desire, mais ils ne peuvent en aucun cas sauter une pi\'ece de l'adversaire.
\PA.\\SO:C.La tour\\FIN:SO. : la tour de shogi se d\'eplace exactement
comme son homologue
des \'echecs c'est-\'a-dire en ligne droite, d'autant de cases qu'elle le
d\'esire. Comme il n'y a pas de dame au shogi, on consid\'ere g\'en\'eralement
que la tour est la pi\'ece la plus puissante de toutes celles de l'\'echiquier.
\PA.\\SO:C.Le fou\\FIN:SO.: cette pi\'ece se d\'eplace \'egalement comme
un fou dans le jeu d'\'echecs n'importe quel nombre de cases en diagonale.
\PA.\\SO:C.Le pion\\FIN:SO. : comme aux \'echecs,
la position initiale de shogi
comporte une rang\'ee de pions qui barre la largeur de l'\'echiquier. Chaque
joueur commence la partie avec neuf pions qui se d\'eplacent vers l'avant. Il
n'y a pas de d\'eplacement double lors du premier coup ; il n'y a pas
de prise en diagonale et il n'y a pas de prise en passant.
\\CR.Pour r\'esumer voici une liste des pi\'eces :
\LISTE.le pion
\\\le cavalier
\\\la tour
\\\le lancier
\\IMPL : ET.\\\le g\'en'eral d'or
\\\le g\'en\'eral d'argent
\\\le roi
\CH::2.
\EN.Attention aux pi\'eces promues \'
\PA.Tandis qu'aux \'echecs seuls les pions
peuvent \^etre promus et deviennent des pi\'eces de valeur sup\'erieure,
l'inter\^et du shogi provient de ce que certaines autres pi\'eces peuvent
\'egalement \^etre promues. On r\'ealise un coup de promotion en d\'eplacant
en partie ou compl\'etement dans la zone de promotion de l'adversaire (les
trois derni\'eres rang\'ees les plus \'eloign\'ees de vous). La promotion a
lieu \'a la fin d'un tel coup mais il n'est pas toujours obligatoire de
promouvoir au shogi. Plusieures pi\'eces peuvent \^etre promues.
\CH::3.
\EN.Cas du g\'en\'eral d'argent
\PA.Une fois promu il se d\'eplace
comme un g\'en\'eral d'or. Sur votre jeu de shogi, le g\'en\'eral d'argent
peut \^etre retourn\'e et sur l'autre cot\'e, vous verrez le symbole d'un
\\SO:C. g\'en\'eral d'argent\\FIN:SO. qui vient d'\^etre promu.
\CH::4.
\EN.Cas des autres pi\'eces
\PA.Le cavalier, le lancier, le pion se d\'eplacent \'egalement comme le
g\'en\'eral d'or une fois qu'ils viennent d'etre promus.
\\CR.La tour lorsqu'elle est promue,
\NO.garde sa capacit\'e originelle de d'\'eplacement \FIN:NO..
d'un nombre quelconque de cases \'a l'horizontale
ou \'a la verticale mais elle acquiert la possibilit\'e de se d\'eplacer
d'une case en diagonale. Le fou, de la m\^eme mani\'ere acquiert la
possibilit\'e de se d\'eplacer d'une case supl\'ementaire \'a l'horizontale
ou \'a la verticale. Si un pion ou un lancier se d\'eplace jusqu'au dernier
rang ou si un cavalier atteint un des deux premiers rangs, la promotion est
obligatoire. Dans tous les autres cas la promotion est optionnelle.
\NO.\\PO:10.il vaut mieux en g\'en\'eral
promouvoir ; cela assure un avantage certain sur
l'adversaire \'a condition qu'il n'ait d\'ej\'a pas eu l'occasion de promouvoir
\CH::1.

\EN.UNE PIECE PRISE AU JEU PEUT ETRE REMISE EN JEU

\PA.Si un joueur d\'eplace une de ses pi\'eces sur une case occup\'ee par l'adversaire, cette derni\'ere est captur\'ee comme aux \'echecs. Mais c'est la diff\'erence de prise qui apporte un inter\^et nouveau \'a ce jeu. Aux \'echecs, quand vous prenez une pi\'ece adverse, elle quitte d\'efinitivement l'\'echiquier pour tout le restant de la partie. Au shogi, vous garder cette pi\'ece en r\'eserve et plus tard dans la partie, vous pourrez la parachuter sur toute case libre (moyennant quelques restrictions). Le parachutage remplace le d\'eplacement d'une pi\'ece d'une case \'a une autre. Une pi\'ece ne peut être que parachut\'ee que non promue, m\^eme si elle avait \'et\'e promue avant sa capture. Quand vous parachutez une pi\'ece prise sur l'\'echiquier, elle vous appartient. La prise d'une pi\'ece ennemie a donc un double int\'er\^et. Un des aspects du parachutage est que vous pouvez tout \'a fait sacrifier une pi\'ece de valeur dans une case contre une pi\'ece de valeur inf\'erieure, simplement parce que vous voulez \^etre capable de parachuter cette derni\'ere sur une autre case dans les quelques coups \'a venir.

\CH::1.\EN.COMME AUX ECHECS, IL FAUT METTRE LE ROI MAT

\PA.Quand un roi est attaqu\'e, il est dit "en \'echec", tout comme aux \'echecs, et le joueur en \'echec doit s'en soustraire imm\'ediatement, en d\'eplacant son propre roi, en capturant la pi\'ece qui provoque cet \'echec ou en interposant une pi\'ece entre les deux. Si un roi est attaqu\'e et qu'il n'y a aucun moyen de le sauver, le joueur a \'et\'e mis en \'echec et mat. Les pi\'eces \'etant toutes en jeu pendant la partie, il est tr\'es rare qu'une partie de shogi se termine sur un nul. Aux \'echecs le nombre de pi\'eces sur l'\'echiquier se reduit durant la partie et la partie risque de se terminer plus souvent par une partie nulle. Ceux qui trouvent ennuyeuses les parties de ma\^itres en \'echecs parce qu'elles sont trop souvent nulles (55 % au plus), n'ont rien \'a craindre au shogi.

\CH::2.

\EN.Comment programmer le shogi ?

\PA.\\CE.On peut appliquer au shogi la plupart des principes de la programmation du jeu d'\'echecs. On commence par d\'evelopper et explorer un arbre de jeu, le probl\'eme essentiel \'etant l'\'enorme facteur de croissance caus\'e par le grand nombre de pi\'eces (40 au lieu d'un maximum de 32) et la possibilit\'e d'un parachutage. Si vous n'avez "en reserve" aucun seul type de pi\'eces captur\'ees, il y aura 42 cases ou plus sur lesquelles on pourra la parachuter. On voit que le nombre de coups l\'egaux disponibles peut facilement aller jusqu'\'a 150 ou 200, d\'es que deux ou trois pi\'eces ennemies ont \'et\'e prises. Il est clair qu'il faut trouver un moyen quelconque d'en r\'eduire le nombre et produire une liste maniable de coups plausibles. On doit dans ce cas utiliser des raisonnements heuristiques intelligents propres au Shogi, connus par les adeptes sous le nom de "proverbes".

\PA.Ceux qui sont int\'eress\'es par l'\'ecriture d'un programme d'\'echecs n'ont qu'\'a se reporter \'a l'\'enorme litt\'erature existante pour d\'ecouvrir des raisonnements heuristiques qui peuvent \^etre employ\'es dans un g\'en\'erateur de coups plausibles ou dans un m\'ecanisme d'\'evaluation. On a aussi \\SO:C.beaucoup \'ecrit sur le Shogi, mais par malchance, presque tout est publi\'e en japonais, \\FIN:SO.

et si votre japonais est aussi mauvais que le mien, vous h\'esiterez \'a faire un safari dans des volumes remplis de symboles myst\'erieux. Je manque de symboles myst\'erieux. Je manque de place pour traiter ces divers raisonnements heuristiques. D'ailleurs, je vous recommande de jeter un coup d'oeil au dos du livre de Fairbain ; il existe

une nombreuse litt\'erature \'a ce sujet.

\PA.Par ailleurs, ceux qui d\'esirent rendre leur programme le plus fort possible devraient s'inscrire \'a l'association de Shogi et essayer d'obtenir les num\'eros du magazine "Shogi" o\'u les principaux proverbes sont expliqu\'es. D\'es que vous avez compris un proverbe, c'est chose facile que de le convertir sous forme num\'erique de fa\,con \'a l'int\'egrer au m\'ecanisme d'\'evaluation de la plausibilit\'e.

\CH::2.

\EN.Deux types d'ouverture sont possibles.

\PA.L'ordre exact dans lequel les coups d'ouverture sont jou\'es n'est pas aussi probl\'ematique au Shogi qu'aux \'echecs. Le plus important dans l'ouverture au Shogi r\'eside dans les cases sur lesquelles on met ses pi\'eces et non dans l'ordre exact qui les y am\'ene . Puisqu'il n'est pas n\'ecessaire que \\FO:G.votre programme de\\POL : 14. Shogi\\FIN : PO. ait acc\'es \'a d'importantes tables\\FIN:FO.
de variantes d'ouvertures, il vous suffit de pr\'eparer une m\'ethode qui incite le programme \'a jouer des coups qui am\'eneront les pi\'eces sur les bonnes cases. Une m\'ethode simple consiste \'a examiner chacune des pi\'eces d'une formation d\'esir\'ee et \'a d\'eterminer, \'a un moment donn\'e de combien de coups chacune d'entre elles est distante de sa case d'arriv\'ee et non dans l'ordre exact qui les y am\'ene .

DESCRIPTION ET UTILISATION DE MACROS

9. DESCRIPTION ET UTILISATION DE MACROS

L'utilisation de macros permet à un utilisateur de se créer un nouveau jeu de commandes. Un jeu permet de remplacer des chaînes de caractères se trouvant dans le texte à formater par de nouvelles chaînes.

Ce chapitre comprendra :

- la description d'une macro
- la procédure à suivre pour tester un jeu de macros
- l'explication des erreurs détectées lors du remplacement
- des exemples d'utilisation
- un conseil aux utilisateurs.

9.1. Description d'une macro

Une macro aura le format :

"<CHAINE REMPLACEE>" = "<CHAINE REMPLACANTE>"

où

<CHAINE REMPLACEE> correspond à une suite d'au plus 10 caractères, chaque occurrence de cette chaîne - dans le texte source - sera remplacée par la **<CHAINE REMPLACANTE>**

<CHAINE REMPLACANTE> correspond à une suite d'au plus 500 caractères, chaque occurrence de la **<CHAINE REMPLACEE>** - dans le texte source - sera remplacée par cette chaîne

9.2. Test d'un jeu de macros

Il convient de tester un jeu de macros avant toute utilisation du formateur.Pour tester un jeu, l'utilisateur devra utiliser la commande :

@MACRO <FICHIER 1> <FICHIER 2> <FICHIER 3>

où

<FICHIER 1> : nom du fichier contenant le texte comprenant les chaînes à remplacer

<FICHIER 2> : nom du fichier contenant les jeux de macros

<FICHIER 3> : nom du fichier qui contiendra les chaînes remplaçantes

9.3. Erreurs détectées

Trois erreurs peuvent être détectées :

ERREUR 1
Une chaîne remplacée a plus de 10 caractéres
Le numéro de la macro erronée sera indiqué

ERREUR 2
Une chaîne à remplacer a plus de 500 caractéres
Le numéro de la macro erronée sera indiqué

ERREUR 3
On n'a pas rencontré un nombre correct de caractéres double kots.

Toute erreur entraîne automatiquement l'arrêt du système de remplacement.

9.4. Exemples

9.4.1. Introduction

Cette partie comprendra un exemple d'utilisation de macros. Cet exemple comprendra trois macros :

1. la définition d'un type de paragraphe particulier
   Ce paragraphe aura des marges droite et gauche redéfinies, la fonte utilisée sera la fonte italique.
2. définition d'un type de liste particulier
   Cette liste aura un décalage plus important que celui de la liste prédéfinie et sera implémentée avec un tiret.
3. définition du header du document
   Le header du document comprendra une demande pour avoir une table des matières à deux niveaux, une pagination, une intialisation des numéros des figures et des pages ainsi que la séquence qui détermine le début du texte à formater

Nous présenterons ces exemples en trois parties :

1. la première partie correspondra au contenu du fichier de macros
   (fichier correspondant au fichier <FICHIER 2> du point 8.2)
2. la deuxième partie correspondra au contenu d'un fichier source qui utilise des appels de macros
   (fichier correspondant au fichier <FICHIER 1> du point 8.2)
3. la troisième partie correspondra au contenu du fichier résultat
   (fichier correspondant au fichier <FICHIER 3> du point 8.2)

9.4.2. Contenu du fichier contenant les macros.

```
"@P1" = "\PA .\\MAR : G : 25.\\MARG : D : 25.\\FO : ITAL."
"@L1" = "\LI .\\IMPL : PO.\\DEC : 20."
"@HEADER" = "\\PAGINATION.\\TABLE : 2.\\NUM : FIG : 3.\\NUM : PAG : 5.
                \\DOCUM."
```

9.4.3. Contenu du fichier source.

```
@HEADER
\titre.Ceci est un exemple de texte comprenant des appels à des macros
\ch : n : 1.
\en.Chapitre de niveau 1.
@P1Ceci est un paragraphe défini au sein d'une macro
\pa.Ceci est un paragraphe "classique"
\lis.Premier élément d'une liste "classique"
\\\Second élément d'une liste "classique"
@L1Premier élément d'une liste définie au sein d'une macro
\\\Second élément d'une lsite définie au sein d'une macro
```

9.4.4. Contenu du fichier résultat

```
\\PAGINATION.\\TABLE : 2.\\NUM : FIG : 3.\\NUM : PAG : 5.
        \\DOCUM.
\titre.Ceci est un exemple de texte comprenant des appels à des macros
\ch : n : 1.
\en.Chapitre de niveau 1.
\PA .\\MAR : G : 25.\\MARG : D : 25.\\FO : ITAL.Ceci est un paragraphe défini au
sein d'une macro
\pa.Ceci est un paragraphe "classique"
\lis.Premier élément d'une liste "classique"
\\\Second élément d'une liste "classique"
\LI .\\IMPL : PO.\\DEC : 20.Premier élément d'une liste définie au sein d'une macro
\\\Second élément d'une liste définie au sein d'une macro
```

9.5. Conseil

Toute occurrence d'une <CHAINE A REMPLACER> étant remplacée par la <CHAINE REMPLACANTE>, il est dangereux de définir une <CHAINE A REMPLACER> sans prendre des précautions. Nous proposons d'utiliser toujours comme premier caractère d'une < CHAINE A REMPLACER> , un caractère qui ne se trouve pas dans le texte source, et qui n'est pas le boa (utilisé comme caractère significatif par le formateur FTTL) ; nous proposons d'utiliser un caractère comme le caractère '@' ou le '%'.

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX (NAMUR)

INSTITUT D'INFORMATIQUE

REALISATION D'UN FORMATEUR (annexes)

mémoire présenté par

Claude Mathieu

et

Alain Schmitz

en vue de l'obtention

du titre de

Licencié et maître en informatique

Année académique 1982-1983

PREFACE

Le but de ce rapport est de présenter l'ensemble des caractéristiques techniques du formateur réalisé.

L'annexe A, complément à l'analyse organique, présente les spécifications de toutes les procédures.

L'annexe B présente l'ensemble des procédures PASCAL.

L'annexe C présente une description du format "FDD".

L'annexe D présente les spécifications et les procédures PASCAL des programmes de décodage/encodage.

L'annexe E présente les caractéristiques techniques de l'imprimante laser CANON LBP10.

L'annexe F présente quelques exemples de feuilles imprimées sur l'imprimante laser CANON LBP10.

L'annexe G présente le programme du pré-processeur de macros.

Chaque annexe se présente sous la forme suivante :

- une page d'entête reprenant le titre de l'annexe
- une page d'introduction non numérotée décrivant le contenu de l'annexe
- le contenu de l'annexe

ANNEXE A

COMPLEMENT

A

L'ANALYSE ORGANIQUE

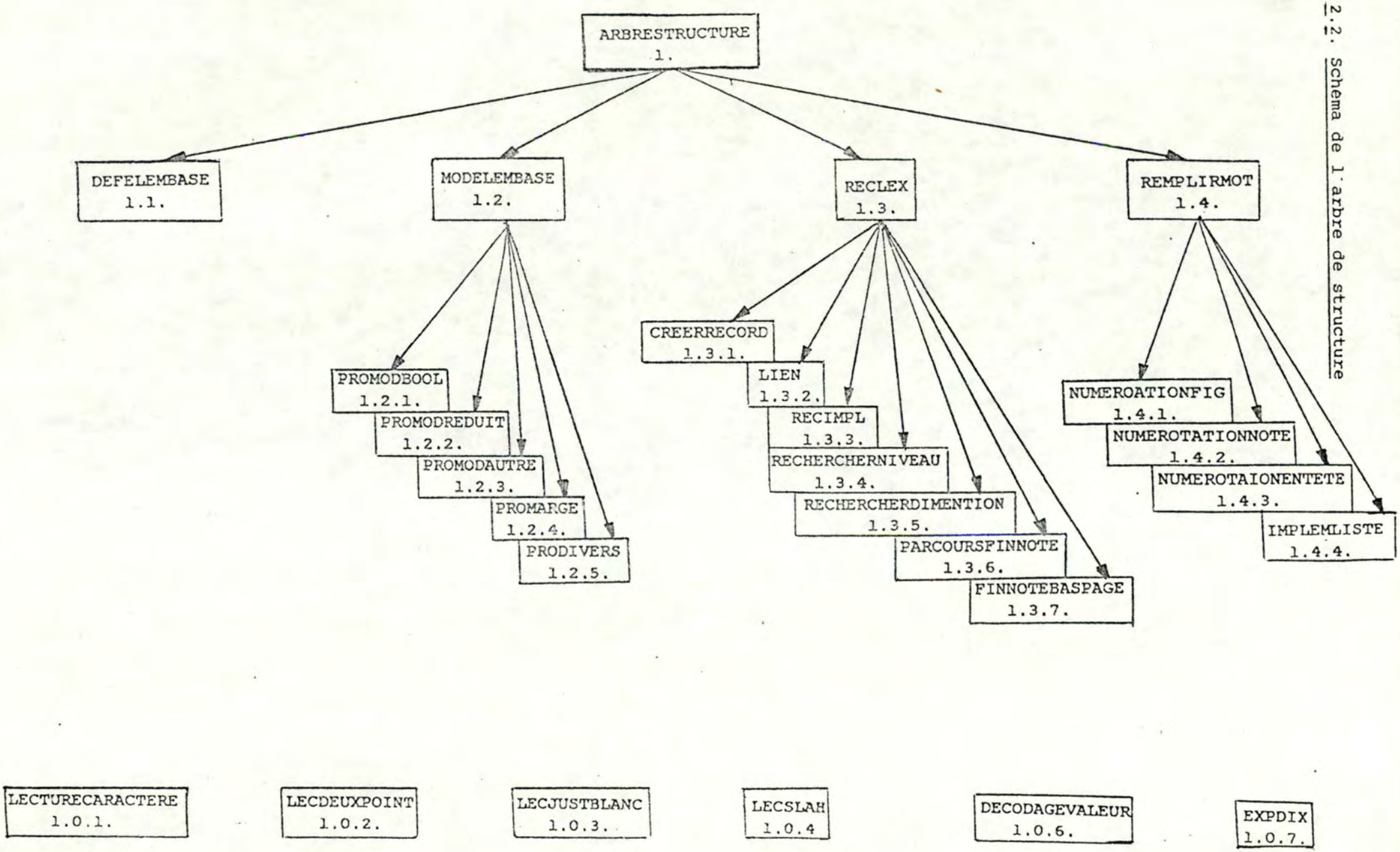
INTRODUCTION

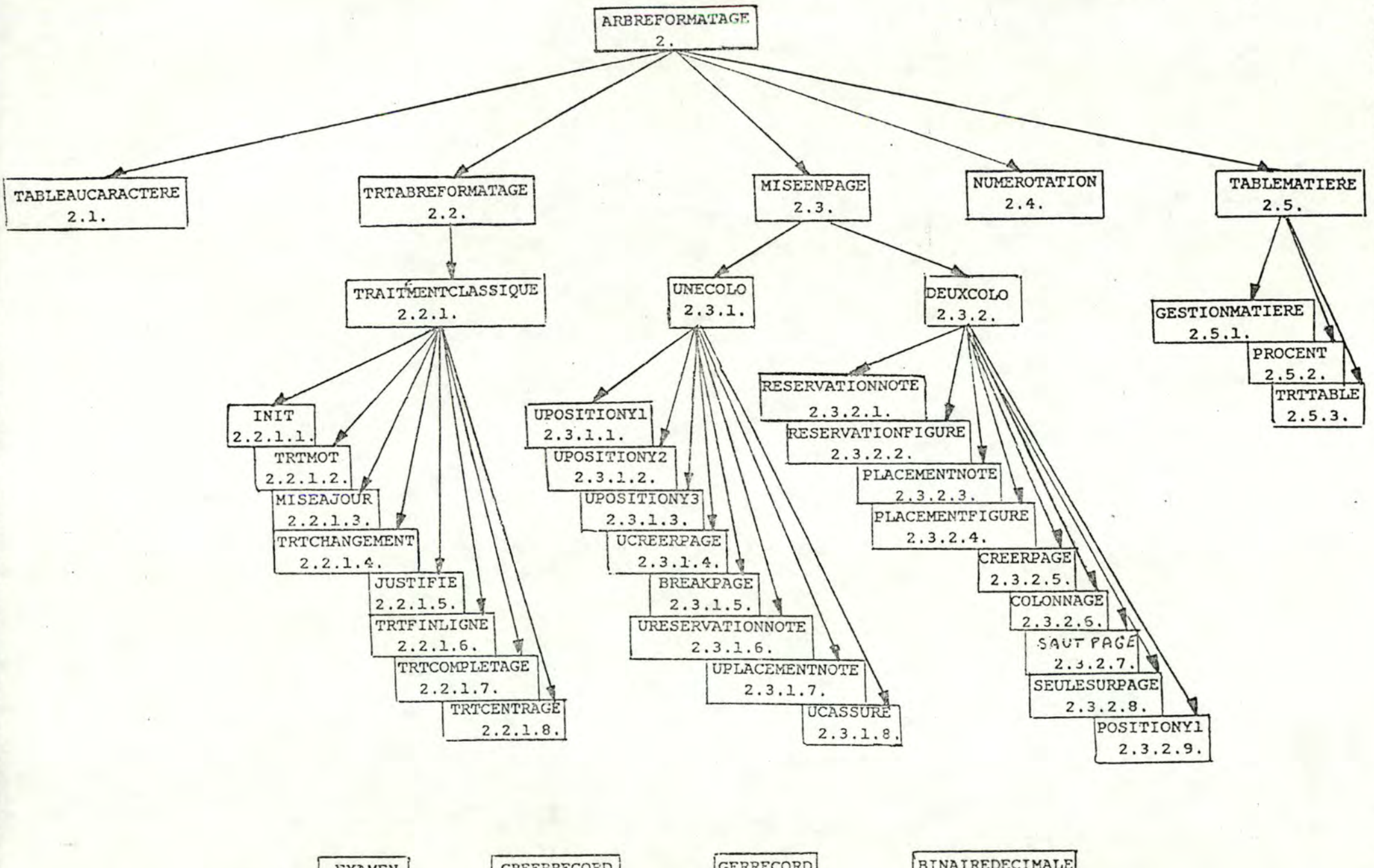
Cette annexe comprend le complément a l'analyse organique qui se compose de :

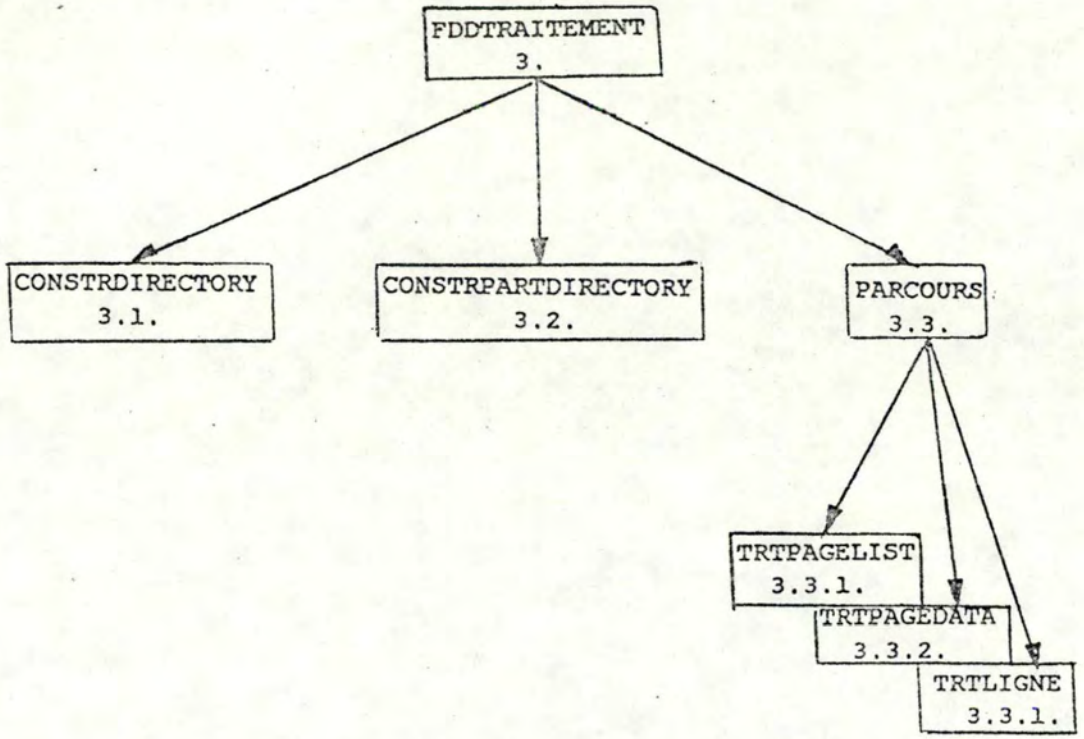
- le plan de l'architecture physique du système qui se compose de la liste des noms des procédures avec leur numéro de page
- le schéma de l'architecture physique (y compris le schéma global). Les numéros sous le nom des procédures font référence aux numéros des procédures au sein des spécifications
- la spécification des procédures

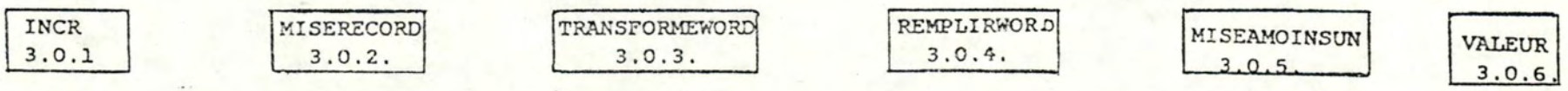
1. PLAN DE L'ARCHITECTURE PHYSIQUE

2.1. Schema global


ARBRESTRUCTURE
1.
DEFELEMBASE
1.1.
MODELEMBASE
1.2.
RECLEX
1.3.
REMPLIRMOT
1.4.
PROMODBOOL
1.2.1.
PROMODREDUIT
1.2.2.
PROMODAUTRE
1.2.3.
PROMARGE
1.2.4.
PRODIVERS
1.2.5.
CREERRECORD
1.3.1.
LIEN
1.3.2.
RECIMPL
1.3.3.
RECHERCHERNIVEAU
1.3.4.
RECHERCHERDIMENTION
1.3.5.
PARCOURSFINNOTE
1.3.6.
FINNOTEBASPAGE
1.3.7.
NUMEROATIONFIG
1.4.1.
NUMEROTATIONNOTE
1.4.2.
NUMEROTAIONENTETE
1.4.3.
IMPLEMLISTE
1.4.4.
LECTURECARACTERE
1.0.1.
LECDEUXPOINT
1.0.2.
LECJUSTBLANC
1.0.3.
LECSLAH
1.0.4
DECODAGEVALEUR
1.0.6.
EXPDIX
1.0.7.


ARBREFORMATAGE
2.
TABLEAUCARACTERE
2.1.
TRTABREFORMATAGE
2.2.
MISEENPAGE
2.3.
NUMEROTATION
2.4.
TABLEMATIERE
2.5.
TRAITMENTCLASSIQUE
2.2.1.
UNECOLO
2.3.1.
DEUXCOLO
2.3.2.
GESTIONMATIERE
2.5.1.
PROCENT
2.5.2.
TRTTABLE
2.5.3.
INIT
2.2.1.1.
TRTMOT
2.2.1.2.
MISEAJOUR
2.2.1.3.
TRTCHANGEMENT
2.2.1.4.
JUSTIFIE
2.2.1.5.
TRTFINLIGNE
2.2.1.6.
TRTCOMPLETAGE
2.2.1.7.
TRTCENTRAGE
2.2.1.8.
UPOSITIONY1
2.3.1.1.
UPOSITIONY2
2.3.1.2.
UPOSITIONY3
2.3.1.3.
UCREERPAGE
2.3.1.4.
BREAKPAGE
2.3.1.5.
URESERVATIONNOTE
2.3.1.6.
UPLACEMENTNOTE
2.3.1.7.
UCASSURE
2.3.1.8.
RESERVATIONNOTE
2.3.2.1.
RESERVATIONFIGURE
2.3.2.2.
PLACEMENTNOTE
2.3.2.3.
PLACEMENTFIGURE
2.3.2.4.
CREERPAGE
2.3.2.5.
COLONNAGE
2.3.2.6.
SAUT PAGE
2.3.2.7.
SEULESURPAGE
2.3.2.8.
POSITIONY1
2.3.2.9.
EXAMEN
2.0.0.1.
CREERRECORD
2.0.0.2.
GERRECORD
2.0.0.3.
BINAIREDECIMALE
2.0.0.4.


FDDTRAITEMENT
3.
CONSTRDIRECTORY
3.1.
CONSTRPARTDIRECTORY
3.2.
PARCOURS
3.3.
TRTPAGELIST
3.3.1.
TRTPAGEDATA
3.3.2.
TRTLIGNE
3.3.1.


INCR
3.0.1
MISERECORD
3.0.2.
TRANSFORMEWORD
3.0.3.
REMPLIRWORD
3.0.4.
MISEAMOINSUN
3.0.5.
VALEUR
3.0.6.

3.SPECIFICATION DES MODULES

3.1. INTRODUCTION

Pour chaque procédure, nous définissons les éléments en entrée, les éléments en sortie, les pré-conditions, les post-conditions, les procédures appelantes et appelées, l'algorithme logique et la spécification des variables.

- les éléments en entrée et les éléments en sortie reprennent la liste des paramètres utilisés dans la procédure ainsi que leur définition.

- les pré-conditions sont les conditions sur les éléments d'entrée des procédures qui doivent être satisfaites pour que toute exécution de la procédure soit correcte.

- les post-conditions sont les conditions qui doivent être satisfaites après toute exécution de la procédure et qui décrivent l'effet attendu de la procédure.

- le(s) procédure(s) appelante(s) sont le(s) procédure(s) qui utilise(nt) la procédure courante

- le(s) procédures(s) appelée(s) sont le(s) procédure(s) utilisée(s) par la procédure courante

- l'algorithme logique exprime la logique de la procédure

- la spécification des variables reprend pour chaque procédure le nom symbolique des variables internes utilisées ainsi que leur définition

Remarques : Toutes les pré- et post-conditions ne sont pas reprises pour chaque procédure. Nous n'avons repris que les pré- et post-conditions de la procédure courante.

Les variables globales ainsi que les fichiers ne sont pas repris à chaque niveau.

Ces conventions ont été prises afin de ne pas alourdir les spécifications.

3.2. SPECIFICATION

PROGRAMME PRINCIPAL

- Entrees :

R10,R11 : fichiers contenant la grandeur des differents jeux de caracteres
SOURCE : fichier contenant le texte au kilometre

- Sorties :

DEFINI : fichier contenant le resultat du formatage (binaire-fdd)
ERREURS : fichier contenant la liste des erreurs faites dans les commandes de formatage

- Effet :

Le programme de formatage a comme but, partant d'un texte au kilometre contenu dans un fichier, de creer un autre fichier contenant toutes les indications pour l'impression sur une imprimante page - cela dans un format etabli (format FDD)

- Procedures appelees :

ARBRESTRUCTURE
PARCOURSARBSTRUC
ARBREFORMATAGE
PARCOURSARBFORM
FDDTRAITEMENT
IMPRBIT

- Algorithme logique :

```
debut
   initialisation des variables ;
   construction de l'arbre de structure;
   construction de l'arbre de formatage ;
   construction du format FDD ;
   decodage et inscription dans un fichier ;
fin
```

- Specification des :

a. Constantes :

PROPORTIONBLANC : valeur donnee au caractere blanc, par rapport au caractere 'i' de la ligne courante (exemple, si le 'i' vaut 3 mm de largeur, alors le caractere blanc vaudra 2 * 3mm, c'est a dire 6mm)
NBPOINTMM : nombre de points par millimetres
ROGNAGE : parametre de rognage pour centrer la feuille

b. Types

TYPEFONTE : liste des genres de fonte disponible
TYPESOULIGNEMENT : liste des genres de soulignement disponible
TYPELISTE : liste des genres de liste disponible
TYPEFEUILLE : liste des types de record de l'arbre de structure
TYFOR : liste des types de record de l'arbre de formatage
GENREFEUILLE : liste des genres de feuilles a imprimer disponible
PMOT : pointeur vers un element secondaire de l'arbre de structure
PGENERAL : pointeur vers un record de l'arbre de structure
PTYPOGRAPHIQUE : pointeur vers un record de l'arbre de formatage
TEX : declaration du type des fichiers
ENSEMBLE : type correspondant a une ligne de caracteres
TAB : tableau de pointeurs vers des chapitres(4 niveaux)

GENERAL : description du record de l'arbre de structure
.CENTRAGE - indication que le record est centre
.MARGEGAUCHE - nombre de millimetres de la marge gauche du record
.MARGEDROITE - nombre de millimetres de la marge droite du record
.MRGMAT - memorisation de la marge gauche/pour modification locale
.MRDMAT - memorisation de la marge droite/pour modification locale
.SEULE - indication que les elements d'identification doivent se trouver seuls sur ne page
.MARGEINFERIEURE - nombre de millimetres de la marge inferieure ; indication valable pour tout le document
.MARGESUPERIEURE - nombre de millimetres de la marge superieure ; indication valable pour tout le document
.LARGEURFEUILLE - nombre de millimetres en largeur des feuilles pour lesquelles un formatage est realise
.LONGUEURFEUILLE - nombre de millimetres en hauteur des feuilles pour lesquelles un formatage est realise
.HAUTEUR - nombre de millimetres a reserver en hauteur pour un record de type figure
.LARGEUR - nombre de millimetes a reserver en largeur pour un record de type figure
.POLAN - memorisation du type de police / pour modification locale
.POLTAB - type de police pour une table des matieres
.POLICE - type de police du record
.NIVEAUCHAPITRE - numero du niveau du record de type chapitre
.DECALAGEPARAGRAPHE - nombre de millimetres a laisser pour decaler la premiere ligne d'un paragraphe
.FOTAB - type de fonte pour la table des matieres
.FONAN - memorisation du type de fonte / pour modification locale
.FONTE - type de fonte du record
.INTERTAB - nombre de millimetres a laisser entre deux elements de la table des matieres
.INTERENITEDOCU - nombre de millimetres moyen a laisser entre

deux entites

.INTERLIGNE - nombre de millimètres à laisser entre deux lignes du record
.INDEX - booléen indiquant que le document doit comporter ou non un index
.TABLEMATIERE - booléen indiquant que le document doit ou non comporter une table des matières
.COLONNAGE - booléen indiquant que le document doit être réalisé ou non en double colonnage
.PAGINATION - booléen indiquant que le document doit ete ou non paginé
.SUPERIEURE - booléen indiquant que le numéro des pages sera ou non au dessus de la page
.SOUTAB - type de soulignement pour la table des matières
.SOUAN - mémorisation du type de soulignement / pour modification locale
.SOULIGNEMENT - type de soulignement désiré pour le record
.IMPRREDUITE - booléen indiquant qu'un nombre limités de feuilles pour l'impression est demandé
.PREMIERE - indicateur du numéro de la première feuille à imprimer
.DERNIERE - indicateur du numéro de la dernière feuille à imprimer
.TFEUILLE - indicateur du type de feuille pour lequel le formatage devra être réalisé
.LISTE - type d'implémentation désirée pour le record de type liste
.TYFEUILLE - type du record de l'arbre de structure
.SAUTPAGE - booléen indiquant si le record courant doit être imprimé sur la page suivante
.SAUTLIGNE - booléen indiquant que les caractères suivants du record courant doivent être sur la ligne suivante
.POINTSUCCESS1 - pointeur vers un record GENERAL
.POINTSUCCESS2 - pointeur vers un record GENERAL
.POINTSUIVANT - pointeur vers un record GENERAL
.POINTMOT - pointeur vers un record MOT

MOT : description du record de l'arbre de structure de type "secondaire", contenant les caractères liés au record de type GENERAL
.SOULIGNEMENT - type de soulignement désiré
.POLICE - type de police désirée
.STRING - caractères du mot
.FONTE - type de fonte désirée
.POINTMOT - pointeur vers le mot suivant
.SAUTLIGNE - booléen indiquant que le mot doit être imprimé sur la ligne suivante

TYPOGRAPHIQUE : description du record de l'arbre de formatage
.TYPESORTE - type du record typographique (page,entite,ligne,bout de ligne)
.TYPO - type physique du record (pargraphe, ...)
.X - nombre de points horizontalement par rapport au record englobant
.Y - nombre de points verticalement par rapport au record englobant
.DELTAX - nombre de points horizontalement du record courant
.DELTAY - nombre de points verticalement du record courant
.POLICETYPOGRAPHIQUE - type de police du record
.FONTETYPOGRAPHIQUE - type de fonte du record
.SOULIGNEMENTTYPOGRQPHIQUE - type de soulignement du record
.STRINGTYPOGRAPHIQUE - caractères du record
.NOMBREBLANC - nombre de caractères blancs du record
.LARGEURVOULUE - largeur désirée pour le record
.NOMBREELEMENT -
. pour un record de type
- page : nombre d'entités contenues
- entite : nombre de lignes contenues
- ligne : nombre de bouts de ligne contenus
- bout de ligne : nombre de caractères
.VALUESPACE - nombre de points du caractère blanc
.INTERLIGNEENTITE - nombre de points entre deux entitées
.SAUTPAGETYPOGRAPHIQUE - booléen indiquant que le record doit ete imprimé sur la page suivante
.POINTSUCCESS - pointeur vers un record TYPOGRAPHIQUE de même type
.POINTSUIVANT - pointeur vers un record TYPOGRAPHIQUE "inférieure"

PTABRECORD : pointeur vers un bloc de 512 bytes
ZEROUN : valeur 0 ou 1
BYTE : ensemble de 8 bits
WORD : ensemble de 2 bytes
BLOCK : ensemble de 206 words ou de 512 bytes
TABRECOR : record contenant les indications de formatage en fomat binaire
.ZONETRAVAIL - élément de type bloc contenant les informations
.POINTSUIVANT - pointeur vers le TABRECOR suivant
.NUMERO - numéro du bloc dans la chaine des blocs

deux entites
.INTERLIGNE - nombre de millimètres à laisser entre deux lignes du record
.INDEX - booléen indiquant que le document doit comporter ou non un index
.TABLEMATIERE - booléen indiquant que le document doit ou non comporter une table des matières
.COLONNAGE - booléen indiquant que le document doit être realise ou non en double colonnage
.PAGINATION - booléen indiquant que le document doit ete ou non paginé
.SUPERIEURE - booléen indiquant que le numéro des pages sera ou non au dessus de la page
.SOUTA3 - type de soulignement pour la table des matieres
.SOUAN - mémorisation du type de soulignement / pour modification locale
.SOULIGNEMENT - type de soulignement désiré pour le record
.IMPRREDUITE - booléen indiquant qu'un nombre limités de feuilles pour l'impression est demandé
.PREMIERE - indicateur du numéro de la premiere feuille à imprimer
.DERNIERE - indicateur du numéro de la derniere feuille à imprimer
.TFEUILLE - indicateur du type de feuille pour lequel le formatage devra etre réalisé
.LISTE - type d'implémentation désirée pour le record de type liste
.TYPEUILLE - type du record de l'arbre de structure
.SAUTPAGE - booléen indiquant si le record courant doit être imprimé sur la page suivante
.SAUTLIGNE - booléen indiquant que les caractères suivants du record courant doivent être sur la ligne suivante
.POINTSUCCESS1 - pointeur vers un record GENERAL
.POINTSUCCESS2 - pointeur vers un record GENERAL
.POINTSUIVANT - pointeur vers un record GENERAL
.POINTMOT - pointeur vers un record MOT

MOT : description du record de l'arbre de structure de type "secondaire", contenant les caractères liés au record de type GENERAL
.SOULIGNEMENT - type de soulignement désiré
.POLICE - type de police désirée
.STRING - caractères du mot
.FONTE - type de fonte désirée
.POINTMOT - pointeur vers le mot suivant
.SAUTLIGNE - booléen indiquant que le mot doit etre imprimé sur la ligne suivante

TYPOGRAPHIQUE : description du record de l'arbre de formatage
.TYPESORTE - type du record typographique (page,entite,ligne,bout de ligne)
.TYPO - type physique du record (pargraphe, ...)
.X - nombre de points horizontalement par rapport au record englobant
.Y - nombre de points verticalement par rapport au record englobant
.DELTAX - nombre de points horizontalement du record courant
.DELTAY - nombre de points verticalement du record courant
.POLICETYPOGRAPHIQUE - type de police du record
.FONTETYPOGRAPHIQUE - type de fonte du record
.SOULIGNEMENTTYPOGRQPHIQUE - type de soulignement du record
.STRINGTYPOGRAPHIQUE - caractères du record
.NOMBREBLANC - nombre de caractères blancs du record
.LARGEURVOULUE - largeur désirée pour le record
.NOMBREELEMENT -
. pour un record de type
- page : nombre d'entités contenues
- entité : nombre de lignes contenues
- ligne : nombre de bouts de ligne contenus
- bout de ligne : nombre de caractères
.VALUESPACE - nombre de points du caractère blanc
.INTERLIGNEENTITE - nombre de points entre deux entitées
.SAUTPAGETYPOGRAPHIQUE - booléen indiquant que le record doit ete imprimé sur la page suivante
.POINTSUCCESS - pointeur vers un record TYPOGRAPHIQUE de même type
.POINTSUIVANT - pointeur vers un record TYPOGRAPHIQUE "inférieure"

PTABRECORD : pointeur vers un bloc de 512 bytes
ZEROUN : valeur 0 ou 1
BYTE : ensemble de 8 bits
WORD : ensemble de 2 bytes
BLOCK : ensemble de 206 words ou de 512 bytes
TABRECOR : record contenant les indications de formatage en fomat binaire
.ZONETRAVAIL - élément de type bloc contenant les informations
.POINTSUIVANT - pointeur vers le TABRECOR suivant
.NUMERO - numéro du bloc dans la chaine des blocs

c. Variables

CCENTAUT :
booléen indiquant le centrage pour les éléments de type auteur
CCENTDAT :
booléen indiquant le centrage pour les éléments de type date
CCENTEN1 :
booléen indiquant le centrage pour les éléments de type entête de niveau 1
CCENTEN2 :
booléen indiquant le centrage pour les éléments de type entête de niveau 2
CCENTEN3 :
booléen indiquant le centrage pour les éléments de type entête de niveau 3
CCENTEN4 :
booléen indiquant le centrage pour les éléments de type entête de niveau 4
CCENTLIS :
booléen indiquant le centrage pour les éléments de type liste
CCENTNOT :
booléen indiquant le centrage pour les éléments de type note bas de page
CCENTPAR :
booléen indiquant le centrage pour les éléments de type paragraphe
CCENTRES :
booléen indiquant le centrage pour les éléments de type resume
CCENTTIT :
booléen indiquant le centrage pour les éléments de type titre
CCENTTAB :
booléen indiquant le centrage pour la table des matières
CCOLODOC :
indicateur booléen indiquant que l'utilisateur veut un double colonnage
CDECPARA :
grandeur en mm de l'espace vide laissé en première ligne d'un paragraphe
CDECALIS :
grandeur en mm de l'espace vide à laisser devant une liste
CFONAUTE :
type de la fonte pour les éléments de type auteur
CFONDATE :
type de la fonte pour les éléments de type date
CFONENN1 :
type de la fonte pour les éléments de type entête de niveau 1
CFONENN2 :
type de la fonte pour les éléments de type entête de niveau 2
CFONENN3 :
type de la fonte pour les éléments de type entête de niveau 3
CFONENN4 :
type de la fonte pour les éléments de type entête de niveau 4
CFONLIST :
type de la fonte pour les éléments de type liste
CFONNOTE :
type de la fonte pour les éléments de type note bas de page
CFONPARA :
type de la fonte pour les éléments de type paragraphe
CFONRESU :
type de la fonte pour les éléments de type resume
CFONTITR :
type de la fonte pour les éléments de type titre
CFONTABL :
type de la fonte pour la table des matières
CIMPLIST :
type d'implementation pour la liste

CIMPREDU :
indicateur booléen indiquant que l'on ne désire qu'un certain nombre de page(s) imprimée(s)
CINFERIE :
numéro de la première page à imprimer
CDERNIER :
numéro de la dernière page à imprimer
CINDEXDO :
indication booléen que l'utilisateur désire un index
CINLIAUT :
nombre de mm de l'interligne pour les éléments de type auteur
CINLIDAT :
nombre de mm de l'interligne pour les éléments de type date
CINLIEN1 :
nombre de mm de l'interligne pour les éléments de type entête de niveau 1
CINLIEN2 :
nombre de mm de l'interligne pour les éléments de type entête de niveau 2
CINLIEN3 :
nombre de mm de l'interligne pour les éléments de type entête de niveau 3
CINLIEN4 :
nombre de mm de l'interligne pour les éléments de type entête de niveau 4
CINLILIS :
nombre de mm de l'interligne pour les éléments de type liste
CINLINOT :
nombre de mm de l'interligne pour les éléments de type note bas de page
CINLIPAR :
nombre de mm de l'interligne pour les éléments de type paragraphe
CINLIRES :
nombre de mm de l'interligne pour les éléments de type resume
CINLITIT :
nombre de mm de l'interligne pour les éléments de type titre
CINLITAB :
nombre de mm de l'interligne pour la table des matières
CINTEREN :
nombre de mm à laisser entre deux entités
CLARGEUR :
largeur de la feuille utilisée
CLONGUEUR :
longueur de la feuille utilisée
CMARGAUT :
nombre de mm de la marge gauche pour les éléments de type auteur
CMARGDAT :
nombre de mm de la marge gauche pour les éléments de type date
CMARGEN1 :
nombre de mm de la marge gauche pour les éléments de type entête de niveau 1
CMARGEN2 :
nombre de mm de la marge gauche pour les éléments de type entête de niveau 2
CMARGEN3 :
nombre de mm de la marge gauche pour les éléments de type entête de niveau 3
CMARGEN4 :
nombre de mm de la marge gauche pour les éléments de type entête de niveau 4
CMARGLIS :
nombre de mm de la marge gauche pour les éléments de type liste
CMARGNOT :
nombre de mm de la marge gauche pour les éléments de type note bas de page
CMARGPAR :
nombre de mm de la marge gauche pour les éléments de type paragraphe
CMARGRES :
nombre de mm de la marge gauche pour les éléments de type resume

CMARGTIT :
nombre de mm de la marge gauche pour les éléments de type titre
CMARGTAB :
nombre de mm de la marge gauche pour la table des matières
CMARDAUT :
nombre de mm de la marge droite pour les éléments de type auteur
CMARDDAT :
nombre de mm de la marge droite pour les éléments de type date
CMARDEN1 :
nombre de mm de la marge droite pour les éléments de type entête de niveau 1
CMARDEN2 :
nombre de mm de la marge droite pour les éléments de type entête de niveau 2
CMARDEN3 :
nombre de mm de la marge droite pour les éléments de type entête de niveau 3
CMARDEN4 :
nombre de mm de la marge droite pour les éléments de type entête de niveau 4
CMARDLIS :
nombre de mm de la marge droite pour les éléments de type liste
CMARDNOT :
nombre de mm de la marge droite pour les éléments de type note bas de page
CMARDPAR :
nombre de mm de la marge droite pour les éléments de type paragraphe
CMARDRES :
nombre de mm de la marge droite pour les éléments de type resume
CMARDTIT :
nombre de mm de la marge droite pour les éléments de type titre
CMARDTAB :
nombre de mm de la marge droite pour la table des matières
CMARDSUD :
nombre de mm de la marge superieure pour le document
CMARDIND :
nombre de mm de la marge inférieure pour le document
CNUMFIG :
numero de la figure
CNUMNOT :
numero de la note bas de page
CPAGIDOC :
indication que l'on desire une pagination du document
CPAGISUP :
indication que le numéro de page se trouve au dessus
de la page ou non (variable booléen indiquant le centrageenne)
CPOLAUT :
type de police pour les éléments de type auteur
CPOLDAT :
type de police pour les éléments de type date
CPOLEN1 :
type de police pour les éléments de type entête de niveau 1
CPOLEN2 :
type de police pour les éléments de type entête de niveau 2
CPOLEN3 :
type de police pour les éléments de type entête de niveau 3
CPOLEN4 :
type de police pour les éléments de type entête de niveau 4
CPOLLIS :
type de police pour les éléments de type liste
CPOLNOT :
type de police pour les éléments de type note bas de page
CPOLPAR :
type de police pour les éléments de type paragraphe

CPOLRES :
type de police pour les éléments de type resume
CPOLTIT :
type de police pour les éléments de type titre
CPOLTAB :
type de police pour la table des matières
CSEULPAG :
indication que l'on desire les éléments d'identification
seuls sur une page
CSOUAUT :
type de soulignement pour les éléments de type auteur
CSOUDAT :
type de soulignement pour les éléments de type date
CSOUEN1 :
type de soulignement pour les éléments de type entête de niveau 1
CSOUEN2 :
type de soulignement pour les éléments de type entête de niveau 2
CSOUEN3 :
type de soulignement pour les éléments de type entête de niveau 3
CSOUEN4 :
type de soulignement pour les éléments de type entête de niveau 4
CSOULIS :
type de soulignement pour les éléments de type liste
CSOUNOT :
type de soulignement pour les éléments de type note bas de page
CSOUPAR :
type de soulignement pour les éléments de type paragraphe
CSOURES :
type de soulignement pour les éléments de type resume
CSOUTIT :
type de soulignement pour les éléments de type titre
CSOUTAB :
type de soulignement pour la table des matières
CTABNIV :
nombre de niveau pour la table des matières
CTABDOC :
indication que l'utilisateur desire une table des matières
CTYPEFEUI :
type de feuille pour laquelle le document doit etre formaté

ADRESSEPREMIERRECORD : pointeur vers la racine de l'arbre de structure
PAGETYPOGRAPHIQUE : pointeur vers la première page de l'arbre de formatage
ADRESSE : pointeur vers le premier bloc contenant le format FDD

CSAUTAUT,CSAUTDAT,CSAUTEN1,CSAUTEN2,CSAUTEN3,CSAUTEN4,CSAUTNOT,CSAUTPAR,
CSAUTTAB,CSAUTFIG :
indicateur booleen de saut de page
CCENTFIG :
booleen indiquant le centrage pour les elements de type figure
CINLIFIG :
nombre de mm de l'interligne pour les elements de type figure
CMARGFIG :
nombre de mm de la marge gauche pour les elements de type figure
CMARDFIG :
nombre de mm de la marge droite pour les elements de type figure
CFONFIGU :
type de fonte pour les elements de type figure
CPOLFIG :
type de police pour les elements de type figure
CSOUFIG :
type des soulignement pour les elements de type figure
CNUMPAG :
numero de la premire page du document
CNUMENT :
numero de la premire entete de niveau 1

1. PROCEDURE ARBRESTRUCTURE

- Entrée :

rien

- Sortie :

ADRIERRECORD : adresse du premier record de l'arbre de structure

- Effet :

Cette procédure construit un arbre de structure à partir du contenu du fichier source

- Pre-condition :

rien

- Post-condition :

(
ADRIERRECORD
)

- Procédure appelante

PROGRAMME PRINCIPAL

- Procédures appelées :

DEFELEMBASE
MODELEMBASE
CREERRECORD
REMPLIRMOT
LECTURECARACTERE
RECLEX

- Algorithme logique :

```
debut
  gestion des variables globales ;
  lecture d'un caractère dans le fichier source ;
  création du record de type 'document' ;
  tant que le fichier n'est pas complètement lu
    debut
      si le caractère lu est '\' alors traitement en conséquence
      sinon remplissage des mots ;
      lecture d'un caractère dans le fichier source ;
    fin
fin
```

- Spécification des variables :

Dans cette procédure deux types de variables sont utilisées ;le premier type correspond aux variables définissant le formatage, variables utilisées pour assigner des valeurs aux records de l'arbre de structure et le deuxième reprend les variables utilisées pour la procédure .

1ère partie : variables de formatage

CENTREAUTEUR : indication de centrage pour l'élément : auteur
CENTREDATE : indication de centrage pour l'élément : date
CENTREENTNIV1 : indication de centrage pour l'élément : entete de niveau 1
CENTREENTNIV2 : indication de centrage pour l'élément : entete de niveau 2
CENTREENTNIV3 : indication de centrage pour l'élément : entete de niveau 3
CENTREENTNIV4 : indication de centrage pour l'élément : entete de niveau 4
CENTRELISTE : indication de centrage pour l'élément : liste
CENTRENOTE : indication de centrage pour l'élément : note bas de page
CENTREPARA : indication de centrage pour l'élément : paragraphe
CENTRERESUME : indication de centrage pour l'élément : resume
CENTRETITRE : indication de centrage pour l'élément : titre
CENTRETABLE : indication de centrage pour la table des matières
COLONNAGEDOCU : indicateur booléen que l'utilisateur veut un double colonnage
DECAPARA : grandeur en mm de l'espace vide laissé en première ligne d'un paragraphe
DECALISTE : grandeur en mm de l'espace vide à laisser devant une liste
FONTEAUTEUR : type de la fonte pour l'élément : auteur
FONTEDATE : type de la fonte pour l'élément : date
FONTEENTNIV1 : type de la fonte pour l'élément : entete de niveau 1
FONTEENTNIV2 : type de la fonte pour l'élément : entete de niveau 2
FONTEENTNIV3 : type de la fonte pour l'élément : entete de niveau 3
FONTEENTNIV4 : type de la fonte pour l'élément : entete de niveau 4
FONTELISTE : type de la fonte pour l'élément : liste
FONTENOTE : type de la fonte pour l'élément : note bas de page
FONTEPARA : type de la fonte pour l'élément : paragraphe
FONTERESUME : type de la fonte pour l'élément : resume
FONTETITRE : type de la fonte pour l'élément : titre
FONTETABLE : type de la fonte pour la table des matières
IMPLLISTE : type d'implémentation pour la liste
IMPREDUITE : indicateur booléen indiquant que l'on ne désire qu'un certain nombre de page imprimée
INFERIEURE : numéro de la première page à imprimer
DERNIERE : numéro de la dernière page à imprimer
INDEXDOCU : indication que l'utilisateur désire un index
INLIAUTEUR : grandeur de l'interligne pour l'élément : auteur
INLIDATE : grandeur de l'interligne pour l'élément : date
INLIENTNIV1 : grandeur de l'interligne pour l'élément : entete de niveau 1
INLIENTNIV2 : grandeur de l'interligne pour l'élément : entete de niveau 2
INLIENTNIV3 : grandeur de l'interligne pour l'élément : entete de niveau 3
INLIENTNIV4 : grandeur de l'interligne pour l'élément : entete de niveau 4
INLILISTE : grandeur de l'interligne pour l'élément : liste
INLINOTE : grandeur de l'interligne pour l'élément : note bas de page
INLIPARA : grandeur de l'interligne pour l'élément : paragraphe
INLIRESUME : grandeur de l'interligne pour l'élément : resume
INLITITRE : grandeur de l'interligne pour l'élément : titre
INLITABLE : grandeur de l interligne pour la table des matières
INTERENTITE : nombre de mm à laisser entre deux entites
LARGEURFEUILLEDOCU : largeur de la feuille utilisée

```
LONGUEURFEUILLEDOCU : longueur de la feuille utilisee
MARGEGAUCHEAUTEUR  : grandeur de la marge gauche pour l'element : auteur
MARGEGAUCHEDATE    : grandeur de la marge gauche pour l'element : date
MARGEGAUCHEENTNIV1 : grandeur de la marge gauche pour l'element : entete de niveau 1
MARGEGAUCHEENTNIV2 : grandeur de la marge gauche pour l'element : entete de niveau 2
MARGEGAUCHEENTNIV3 : grandeur de la marge gauche pour l'element : entete de niveau 3
MARGEGAUCHEENTNIV4 : grandeur de la marge gauche pour l'element : entete de niveau 4
MARGEGAUCHELISTE   : grandeur de la marge gauche pour l'element : liste
MARGEGAUCHENOTE    : grandeur de la marge gauche pour l'element : note bas de page
MARGEGAUCHEPARA    : grandeur de la marge gauche pour l'element : paragraphe
MARGEGAUCHERESUME  : grandeur de la marge gauche pour l'element : resume
MARGEGAUCHETITRE   : grandeur de la marge gauche pour l'element : titre
MARGEGAUCHETABLE   : grandeur de la marge gauche pour la table des matieres
MARGEDROITEAUTEUR  : grandeur de la marge droite pour l'element : auteur
MARGEDROITEDATE    : grandeur de la marge droite pour l'element : date
MARGEDROITEENTNIV1 : grandeur de la marge droite pour l'element : entete de niveau 1
MARGEDROITEENTNIV2 : grandeur de la marge droite pour l'element : entete de niveau 2
MARGEDROITEENTNIV3 : grandeur de la marge droite pour l'element : entete de niveau 3
MARGEDROITEENTNIV4 : grandeur de la marge droite pour l'element : entete de niveau 4
MARGEDROITELISTE   : grandeur de la marge droite pour l'element : liste
MARGEDROITENOTE    : grandeur de la marge droite pour l'element : note bas de page
MARGEDROITEPARA    : grandeur de la marge droite pour l'element : paragraphe
MARGEDROITERESUME  : grandeur de la marge droite pour l'element : resume
MARGEDROITETITRE   : grandeur de la marge droite pour l'element : titre
MARGEDROITETABLE   : grandeur de la marge droite pour la table des matieres
MARGESUPDOCU       : grandeur de la marge superieure pour le document
MARGEINTDOCU       : grandeur de la marge inferieure pour le document
NUMEROFIG          : numero de la figure
NUMERONOTE         : numero de la note bas de page
PAGINATIONDOCU     : indication que l'on desire une pagination du document
PAGINATIONSUPERIEURE : indication que le numero de page se trouve au dessus
                       de la page ou non (variable booleenne)
POLICEAUTEUR       : type de police pour l'element : auteur
POLICEDATE         : type de police pour l'element : date
POLICEENTNIV1      : type de police pour l'element : entete de niveau 1
POLICEENTNIV2      : type de police pour l'element : entete de niveau 2
POLICEENTNIV3      : type de police pour l'element : entete de niveau 3
POLICEENTNIV4      : type de police pour l'element : entete de niveau 4
POLICELISTE        : type de police pour l'element : liste
POLICENOTE         : type de police pour l'element : note bas de page
POLICEPARA         : type de police pour l'element : paragraphe
POLICERESUME       : type de police pour l'element : resume
POLICETITRE        : type de police pour l'element : titre
POLICETABLE        : type de police pour la table des matieres
SEULESURPAGEDOCU   : indication que l'on desire les elements d'identification
                     seuls sur une page
SOULIAUTEUR        : type de soulignement pour l'element : auteur
SOULIDATE          : type de soulignement pour l'element : date
SOULIENTNIV1       : type de soulignement pour l'element : entete de niveau 1
SOULIENTNIV2       : type de soulignement pour l'element : entete de niveau 2
SOULIENTNIV3       : type de soulignement pour l'element : entete de niveau 3
SOULIENTNIV4       : type de soulignement pour l'element : entete de niveau 4
SOULILISTE         : type de soulignement pour l'element : liste
SOULINOTE          : type de soulignement pour l'element : note bas de page
SOULIPARA          : type de soulignement pour l'element : paragraphe
SOULIRESUME        : type de soulignement pour l'element : resume
SOULITITRE         : type de soulignement pour l'element : titre
SOULITABLE         : type de soulignement pour la table des matieres
TABLENIV           : nombre de niveau pour la table des matieres
TABLEMATIEREDOCU   : indication que l'utilisateur desire une table des matieres
TYPEFEUILLEDOCU    : type de feuille pour laquelle le document doit etre formate
```

```
SAUTAUTEUR      : indication de saut de page pour l'element : auteur
SAUTDATE        : indication de saut de page pour l'element : date
SAUTENTETE1     : indication de saut de page pour l'element : entête de niveau 1
SAUTENTETE2     : indication de saut de page pour l'element : entête de niveau 2
SAUTENTETE3     : indication de saut de page pour l'element : entête de niveau 3
SAUTENTETE4     : indication de saut de page pour l'element : entête de niveau 4
SAUTFIGURE      : indication de saut de page pour l'element : figure
SAUTLISTE       : indication de saut de page pour l'element : liste
SAUTNOTE        : indication de saut de page pour l'element : note bas de page
SAUTPARAGRAPHE  : indication de saut de page pour l'element : paragraphe
SAUTRESUME      : indication de saut de page pour l'element : resume
SAUTTITRE       : indication de saut de page pour l'element : titre
SAUTTABLE       : indication de saut de page pour l'element : table des matires
CENTREFIGURE    : indicateur de centrage pour l'element : figure
MARGEDROITEFIGURE : grandeur de la marge droite pour l'element : figure
MARGEGAUCHEFIGURE : grandeur de la marge gauche pour l'element : figure
INLIFIGURE      : grandeur de l'interligne de l'element : figure
FONTEFIGURE     : type de fonte de l'element : figure
POLICEFIGURE    : type de police de l'element : figure
SOULIFIGURE     : type des soulignement de l'element : figure
NUMEROPAGE      : numero de la premiére page
NUMEROENTETE    : numero de la premiére entete de niveau 1
```

2ieme partie : variables de la procedure

```
IND1          : variable entiere utilisee pour indiquer le numero d'un chapitre
                ou la largeur d'une figure / sans utilite ici
IND2          : variable entiere utilisee pour indiquer la hauteur d'une figure
                / sans utilite ici
I             : compteur de boucle
IMPL          : booleen  qui indique le passage dans la procedure RECIMPL
FANION        : booleen qui indique la rencontre d' au moins 2 boas dans le fichier
                source
CREER         : booleen qui indique la creation d' un record
FIN           : booleen qui indique l'obtention de la fin du fichier source
TROUVER       : booleen qui indique que la sequence typographique a ete reconnue
IERMOT        : booleen qui indique la création d'un mot pour l'element
                de structure courant
IERPAS        : booleen qui indique le passage dans la procedure
                REMPLIRMOT pour l'element de structure courant
DEJASEP       : booleen qui indique la rencontre d' un caractere
                separateur pour l'element de structure courant
CREERLISTE    : booleen qui indique la création d'une liste
NEWLISTE      : booleen qui indique l'obtention d'un nouvel element pour l'element
                courant de structure de type liste
MOTDEJASLAH   : booleen indiquant que le caractere precedent etait un boa
MEMNEWLISTE   : booleen memorisant la valeur de NEWLISTE
TABLEAU       : tableau contenant le numero courant des entetes
IERRECORD     : pointeur vers le premier record de l'arbre de structure
PREC          : pointeur vers le dernier element de l'arbre de structure
PRECP         : pointeur vers l'avant dernier element de l'arbre de structure
PRECPP        : pointeur de memorisation de la valeur de PRECP
MEMMOT        : pointeur vers le dernier mot du dernier element de l'arbre de structure
CH            : caractere lu dans le fichier source
```

1.1 PROCEDURE DEFELEMBASE

- Entrée :

  rien

- Sortie :

  rien

- Pré-condition :

  rien

- Post-conditions :

  (les variables ci-dessous ont reçus une affectation - valeur définie au niveau global)

(
(CENTREAUTEUR = CCENTAUT) ∧ (CENTREDATE = CCENTDAT)
∧(CENTREENTNIV1 = CCENTEN1) ∧ (CENTREENTNIV2 = CCENTEN2)
∧(CENTREENTNIV3 = CCENTEN3) ∧ (CENTREENTNIV4 = CCENTEN4)
∧(CENTRELISTE = CCENTLIS) ∧ (CENTRENOTE = CCENTNOT)
∧(CENTREPARA = CCENTPAR) ∧ (CENTRERESUME = CCENTRES)
∧(CENTRETITRE = CCENTTIT) ∧ (CENTRETABLE = CCENTTAB)
∧(COLONNAGEDOCU = CCOLODOC) ∧ (DECAPARA = CDECPARA)
∧(DECALISTE = CDECALIS) ∧ (FONTEAUTEUR = CFONAUTE)
∧(FONTEDATE = CFONDATE) ∧ (FONTEENTNIV1 = CFONENN1)
∧(FONTEENTNIV2 = CFONENN2) ∧ (FONTEENTNIV3 = CFONENN3)
∧(FONTELISTE = CFONLIST) ∧ (FONTENOTE = CFONNOTE)
∧(FONTEPARA = CFONPARA) ∧ (FONTERESUME = CFONRESU)
∧(FONTETITRE = CFONTITR) ∧ (FONTETABLE = CFONTABL)
∧(IMPLLISTE = CIMPLIST) ∧ (IMPREDUITE = CIMPREDU)
∧(INTERIEURE = CINTERIE) ∧ (DERNIERE = CDERNIER)
∧(INDEXDOCU = CINDEXDO) ∧ (INLIAUTEUR = CINLIAUT)
∧(INLIDATE = CINLIDAT) ∧ (INLIENTNIV1 = CINLIEN1)
∧(INLIENTNIV2 = CINLIEN2) ∧ (INLIENTNIV3 = CINLIEN3)
∧(INLIENTNIV4 = CINLIEN4) ∧ (INLILISTE = CINLILIS)
∧(INLINOTE = CINLINOT) ∧ (INLIPARA = CINLIPAR)
∧(INLIRESUME = CINLIRES) ∧ (INLITITRE = CINLITIT)
∧(INLITABLE = CINLITAB) ∧ (INTERENTITE = CINTEREN)
∧(LARGEURFEUILLEDOCU = CLARGEUR) ∧ (LONGUEURFEUILLEDOCU = CLONGUEUR)
∧(MARGEGAUCHEAUTEUR = CMARGAUT) ∧ (MARGEGAUCHEDATE = CMARGDAT)
∧(MARGEGAUCHEENTNIV1 = CMARGEN1) ∧ (MARGEGAUCHEENTNIV2 = CMARGEN2)
∧(MARGEGAUCHEENTNIV3 = CMARGEN3) ∧ (MARGEGAUCHEENTNIV4 = CMARGEN4)
∧(MARGEGAUCHELISTE = CMARGLIS) ∧ (MARGEGAUCHENOTE = CMARGNOT)
∧(MARGEGAUCHEPARA = CMARGPAR) ∧ (MARGEGAUCHERESUME = CMARGRES)
∧(MARGEGAUCHETITRE = CMARGTIT) ∧ (MARGEGAUCHETABLE = CMARGTAB)
∧(MARGEDROITEAUTEUR = CMARDAUT) ∧ (MARGEDROITEDATE = CMARDDAT)
∧(MARGEDROITEENTNIV1 = CMARDEN1) ∧ (MARGEDROITEENTNIV2 = CMARDEN2)
∧(MARGEDROITEENTNIV3 = CMARDEN3) ∧ (MARGEDROITEENTNIV4 = CMARDEN4)
∧(MARGEDROITELISTE = CMARDLIS) ∧ (MARGEDROITENOTE = CMARDNOT)
∧(MARGEDROITEPARA = CMARDPAR) ∧ (MARGEDROITERESUME = CMARDRES)
∧(MARGEDROITETITRE = CMARDTIT) ∧ (MARGEDROITETABLE = CMARDTAB)
∧(MARGESUPDOCU = CMARDSUD) ∧ (MARGEINFDOCU = CMARDIND)
∧(NUMEROFIG = CNUMFIG) ∧ (NUMERONOTE = CNUMNOT)
∧(PAGINATIONDOCU = CPAGIDOC) ∧ (PAGINATIONSUPERIEURE = CPAGISUP)
∧(POLICEAUTEUR = CPOLAUT) ∧ (POLICEDATE = CPOLDAT)
∧(POLICEENTNIV1 = CPOLEN1) ∧ (POLICEENTNIV2 = CPOLEN2)
∧(POLICEENTNIV3 = CPOLEN3) ∧ (POLICEENTNIV4 = CPOLEN4)
∧(POLICELISTE = CPOLLIS) ∧ (POLICENOTE = CPOLNOT)
∧(POLICEPARA = CPOLPAR) ∧ (POLICERESUME = CPOLRES)
∧(POLICETITRE = CPOLTIT) ∧ (POLICETABLE = CPOLTAB)
∧(SEULESURPAGEDOCU = CSEULPAG) ∧ (SOULIAUTEUR = CSOUAUT)
∧(SOULIDATE = CSOUDAT) ∧ (SOULIENTNIV1 = CSOUEN1)
∧(SOULIENTNIV2 = CSOUEN2) ∧ (SOULIENTNIV3 = CSOUEN3)
∧(SOULIENTNIV4 = CSOUEN4) ∧ (SOULILISTE = CSOULIS)
∧(SOULINOTE = CSOUNOT) ∧ (SOULIPARA = CSOUPAR)
∧(SOULIRESUME = CSOURES) ∧ (SOULITITRE = CSOUTIT)
∧(SOULITABLE = CSOUTAB) ∧ (TABLENIV = CTABNIV)
∧(TABLEMATIEREDOCU = CTABDOC) ∧ (TYPEFEUILLEDOCU = CTYPEFEUI)
∧(SAUTAUTEUR = CSAUTAUT) ∧ (SAUTDATE = CSAUTDAT)
∧(SAUTENTETE1 = CSAUTEN1) ∧ (SAUTENTETE2 = CSAUTEN1)
∧(SAUTENTETE3 = CSAUTEN3) ∧ (SAUTENTETE4 = CSAUTEN3)
∧(SAUTFIGURE = CSAUTFIG) ∧ (SAUTLISTE = CSAUTLIS)
∧(SAUTNOTE = CSAUTNOT) ∧ (SAUTPARAGRAPHE = CSAUTPAR)
∧(SAUTRESUME = CSAUTRES) ∧ (SAUTTITRE = CSAUTTIT)
∧(SAUTTABLE = CSAUTTAB) ∧ (CENTREFIGURE = CCENTFIG)
∧(FONTEFIGURE = CFONFIG) ∧ (INLIFIGURE = CINLIFIG)
∧(MARGEGAUCHEFIGURE = CMARGFIG) ∧ (MARGEDROITEFIGURE = CMARDFIG)
∧(POLICEFIGURE = CPOLFIG) ∧ (SOULIFIGURE = CSOUFIG)
)

- Procédure appelante :

  ARBRESTRUCTURE

- Procedure appelee :

  rien

- Algorithme logique :

  début
    assignation ;
  fin

- Spécification des variables :

  rien

1.2. PROCEDURE MODELEMBASE

- Entrée :

rien

- Sortie :

rien

- Effet :

Cette procédure a pour but de modifier les caractéristiques de formatage prédéfinies à partir des commandes se trouvant en entête du fichier source.

- Pré-condition :

rien

- Post-conditions :

(
((ERREUR(1) ∨ (ERREUR(7)) ∨ rien) ∨ (ERREUR(143))
∧(N variables globales réinitialisées)
∧(on est positionné dans le fichier source sur le caractère '.
qui suit l'indicateur de début du texte à formater)
)
)

- Procédure appelante :

ARBRSTRUC

- Procédures appelées :

LECTURECARACTERE
PROMODAUTRE
PROMODBOOL
PRODIVERS
PROMARGE
PROMODREDUIT
LECPOINT
ERREUR

- Algorithme logique :

```
début
  lecture de 2 caractères ;
  Si les 2 caractères lus sont différents de '\\' ou de '\d'
  ou de '\D'
    erreur
    sinon
      debut
        tant que les caractères lus sont differents de '\d'
        ou de '\D'
          debut
            analyse des caractères lus afin de déterminer
            le traitement à effectuer ;
            lecture de 2 caractères
          fin
      fin
fin
```

- Spécification des variables :

CH	: caractère lu dans sur le fichier source
TAMPON	: ensemble de deux caractères (caractères permettant l'identification de la variable à modifier)
FIN	: booléen indiquant la fin du fichier source

1.2.1. PROCEDURE PROMODBOOL

- Entrées :

  C1 : premier caractère lu dans le fichier source
  C2 : deuxième caractere lu dans le fichier source

- Sortie :

  rien

- Effet :

  Cette procédure analyse les caractères contenus dans le fichier source afin de déterminer la variable globale pré-définie que l'on désire modifier.
  Promodbool s'occupe de la table des matières, de la pagination, du double colonnage, de l'index.

- Pré-conditions :

  (
  C1∧C2
)

- Post-conditions :

  (
  ((C1 = 't')∨(C1 = 'T'))∧((PAGINATIONDOCU = true)
  ∧(TABLEMATIERE = true)))
  ∨((C1 = 'i')∨(C1 = 'I'))∧(INDEXDOCU = true))
  ∨((C1 = 'c')∨(C1 = 'C'))∧(COLONNAGEDOCU = true))
  ∨((C1 = 'u')∨(C1 = 'U'))∧(SEULESURPAGE = true))
)

- Procédure appelante :

  MODELEMBASE

- Procédure appelée :

  rien

- Algorithme logique :

  début
  test sur C1 afin de déterminer la variable à modifier ;
  lecture jusqu'au caractère '\' suivant
  fin

- Spécification des variables :

  rien

1.2.2. PROCEDURE PROMODREDUIT

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procédure analyse les caractères contenus dans le fichier source afin de réinitialiser les variables globales qui permettent de limiter le nombre de page(s) à imprimer.

- Pré-condition :

  rien

- Post-conditions :

  (
  ((IMPREDUITEDOCU = true) ∧ (PREMIERDOCU = ‹valeur›)
  ∧(DERNIEREDOCU = ‹valeur›)) ∨ (ERREUR (1)) ∨ (ERREUR (145))
)

  ‹valeur› représente un nombre entier lu dans le fichier source.

- Procédure appelante :

  PROMODBOOL

- Procédures appelées :

  LECDEUXPOINT
  LECTURECARACTERE
  DECODAGEVALEUR
  LECSLASH
  ERREUR

- Algorithme logique :

  début
  lecture des caractères jusqu'au ':' ;
  lecture de caractères jusqu'à avoir quelque chose de ‹› ' ' ;
  décodage du numéro de la première page à imprimer ;
  lecture de caractères jusqu'à avoir quelque chose de ‹› ' ' ;
  décodage du numéro de la dernière page à imprimer ;
  lecture des caractères jusqu'au '\' suivant ;
  initialisation des variables pré-définies
  fin

- Spécification des variables :

  CH : caractere lu dans le fichier source
  FIN : booléen indiquant la fin du fichier source
  NB : variable entière résultat du décodage

1.2.3. PROCEDURE PROMODAUTRE

- Entrees :

C1 : premier caractere lu dans le fichier source
C2 : deuxieme caractere lu dans le fichier source

- Sortie :

rien

- Effet :

Cette procedure analyse les caracteres en entree et modifie la ou les variable(s) globale(s) pre-definie(s) correspondant a la sequence de caracteres en entree.
Les variables modifiees concernent : le centrage ou la justification, la grandeur de l'interligne, le type de soulignement, le type de fonte, le type de police.

- Pre-conditions :

(C1 ∧ C2)

- Post-conditions :

La procedure utilise des caracteres du fichier source necessaires pour la determination de la variable globale a modifier.
CAR1 : determine le type d'element de structure ,
CAR2 : determine la valeur a assigner a la variable,
CAR3 : utilise au cas où CAR1 n'est pas suffisant pour determiner le type d'element de structure

```
(
 ((C1 = 'P')
 ∧( (((CAR1 = 'T') ∧ (CAR3 = 'I')) ∧((FONTETITRE   = CAR2) ∨ (ERREUR(30))))
   ∨( (CAR1 = 'D')                 ∧((FONTEDATE    = CAR2) ∨ (ERREUR(31))))
   ∨( (CAR1 = 'R')                 ∧((FONTERESUME  = CAR2) ∨ (ERREUR(32))))
   ∨( (CAR1 = 'A')                 ∧((FONTEAUTEUR  = CAR2) ∨ (ERREUR(33))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '1')) ∧((FONTEENTNIV1 = CAR2) ∨ (ERREUR(34))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '2')) ∧((FONTEENTNIV2 = CAR2) ∨ (ERREUR(35))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '3')) ∧((FONTEENTNIV3 = CAR2) ∨ (ERREUR(36))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '4')) ∧((FONTEENTNIV4 = CAR2) ∨ (ERREUR(37))))
   ∨( (CAR1 = 'F')                 ∧((FONTEFIGURE  = CAR2) ∨ (ERREUR(38))))
   ∨( (CAR1 = 'L')                 ∧((FONTELISTE   = CAR2) ∨ (ERREUR(39))))
   ∨( (CAR1 = 'P')                 ∧((FONTEPARA    = CAR2) ∨ (ERREUR(40))))
   ∨( (CAR1 = 'N')                 ∧((FONTENOTE    = CAR2) ∨ (ERREUR(41))))
   ∨(((CAR1 = 'T') ∧ (CAR3 = 'A')) ∧((FONTETABLE   = CAR2) ∨ (ERREUR(42))))
   ∨( (CAR1 <> 'T','D','R','A','E','F','L','N') ∧ (ERREUR(44))
    )
 )
∨
 ((C1 = 'P')
 ∧((((CAR1 = 'T') ∧ (CAR3 = 'I')) ∧((POLICETITRE   = CAR2) ∨ (ERREUR(45))))
   ∨( (CAR1 = 'D')                 ∧((POLICEDATE    = CAR2) ∨ (ERREUR(46))))
   ∨( (CAR1 = 'R')                 ∧((POLICERESUME  = CAR2) ∨ (ERREUR(47))))
   ∨( (CAR1 = 'A')                 ∧((POLICEAUTEUR  = CAR2) ∨ (ERREUR(48))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '1')) ∧((POLICEENTNIV1 = CAR2) ∨ (ERREUR(49))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '2')) ∧((POLICEENTNIV2 = CAR2) ∨ (ERREUR(50))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '3')) ∧((POLICEENTNIV3 = CAR2) ∨ (ERREUR(51))))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '4')) ∧((POLICEENTNIV4 = CAR2) ∨ (ERREUR(52))))
   ∨( (CAR1 = 'F')                 ∧((POLICEFIGURE  = CAR2) ∨ (ERREUR(53))))
   ∨( (CAR1 = 'L')                 ∧((POLICELISTE   = CAR2) ∨ (ERREUR(54))))
   ∨( (CAR1 = 'P')                 ∧((POLICEPARA    = CAR2) ∨ (ERREUR(55))))
   ∨( (CAR1 = 'N')                 ∧((POLICENOTE    = CAR2) ∨ (ERREUR(56))))
   ∨(((CAR1 = 'T') ∧ (CAR3 = 'A')) ∧((POLICETABLE   = CAR2) ∨ (ERREUR(57))))
   ∨( (CAR1 <> 'T','D','R','A','E','F','L','N') ∧ (ERREUR(58))
    )
 )
∨
 ((C1 = 'C')
 ∧((((CAR1 = 'T') ∧ (CAR3 = 'I')) ∧ (CENTRETITRE   = TRUE))
   ∨( (CAR1 = 'D')                 ∧ (CENTREDATE    = TRUE))
   ∨( (CAR1 = 'R')                 ∧ (CENTRERESUME  = TRUE))
   ∨( (CAR1 = 'A')                 ∧ (CENTREAUTEUR  = TRUE))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '1')) ∧ (CENTREENTNIV1 = TRUE))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '2')) ∧ (CENTREENTNIV2 = TRUE))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '3')) ∧ (CENTREENTNIV3 = TRUE))
   ∨(((CAR1 = 'E') ∧ (CAR3 = '4')) ∧ (CENTREENTNIV4 = TRUE))
   ∨( (CAR1 = 'F')                 ∧ (CENTREFIGURE  = TRUE))
   ∨( (CAR1 = 'L')                 ∧ (CENTRELISTE   = TRUE))
   ∨( (CAR1 = 'P')                 ∧ (CENTREPARA    = TRUE))
   ∨( (CAR1 = 'N')                 ∧ (CENTRENOTE    = TRUE))
   ∨(((CAR1 = 'T') ∧ (CAR3 = 'A')) ∧ (CENTRETABLE   = TRUE))
   ∨( (CAR1 <> 'T','D','R','A','E','F','L','N') ∧ (ERREUR(59))
    )
 )
∨
```

```
∨
((C1 = 'J')
 ∧ ((((CAR1 = 'T') ∧ (CAR3 = 'I')) ∧ (CENTRETITRE   = FALSE))
  ∨( (CAR1 = 'D')                  ∧ (CENTREDATE    = FALSE))
  ∨( (CAR1 = 'R')                  ∧ (CENTRERESUME  = FALSE))
  ∨( (CAR1 = 'A')                  ∧ (CENTREAUTEUR  = FALSE))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '1')) ∧ (CENTREENTNIV1 = FALSE))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '2')) ∧ (CENTREENTNIV2 = FALSE))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '3')) ∧ (CENTREENTNIV3 = FALSE))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '4')) ∧ (CENTREENTNIV4 = FALSE))
  ∨( (CAR1 = 'F')                  ∧ (CENTREFIGURE  = FALSE))
  ∨( (CAR1 = 'L')                  ∧ (CENTRELISTE   = FALSE))
  ∨( (CAR1 = 'P')                  ∧ (CENTREPARA    = FALSE))
  ∨( (CAR1 = 'N')                  ∧ (CENTRENOTE    = FALSE))
  ∨(((CAR1 = 'T') ∧ (CAR3 = 'A')) ∧ (CENTRETABLE   = FALSE))
  ∨( (CAR1 <> 'T','D','R','A','E','F','L','N') ∧ (ERREUR(60))
   )
 )
∨
((C1 = 'I')
 ∧((((CAR1 = 'T') ∧ (CAR3 = 'I')) ∧ ((INLITITRE   = CAR2) ∨ (ERREUR(61))))
  ∨( (CAR1 = 'D')                  ∧ ((INLIDATE    = CAR2) ∨ (ERREUR(62))))
  ∨( (CAR1 = 'R')                  ∧ ((INLIRESUME  = CAR2) ∨ (ERREUR(63))))
  ∨( (CAR1 = 'A')                  ∧ ((INLIAUTEUR  = CAR2) ∨ (ERREUR(64))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '1')) ∧ ((INLIENTNIV1 = CAR2) ∨ (ERREUR(65))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '2')) ∧ ((INLIENTNIV2 = CAR2) ∨ (ERREUR(66))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '3')) ∧ ((INLIENTNIV3 = CAR2) ∨ (ERREUR(67))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '4')) ∧ ((INLIENTNIV4 = CAR2) ∨ (ERREUR(68))))
  ∨( (CAR1 = 'F')                  ∧ ((INLIFIGURE  = CAR2) ∨ (ERREUR(69))))
  ∨( (CAR1 = 'L')                  ∧ ((INLILISTE   = CAR2) ∨ (ERREUR(70))))
  ∨( (CAR1 = 'P')                  ∧ ((INLIPARA    = CAR2) ∨ (ERREUR(71))))
  ∨( (CAR1 = 'N')                  ∧ ((INLINOTE    = CAR2) ∨ (ERREUR(72))))
  ∨(((CAR1 = 'T') ∧ (CAR3 = 'A')) ∧ ((INLITABLE   = CAR2) ∨ (ERREUR(73))))
  ∨( (CAR1 <> 'T','D','R','A','E','F','L','N') ∧ (ERREUR(74))
   )
 )
∨
((C1 = 'S')
 ∧ ((((CAR1 = 'T') ∧ (CAR3 = 'I')) ∧ ((SOULITITRE   = CAR2) ∨ (ERREUR(75))))
  ∨( (CAR1 = 'D')                  ∧ ((SOULIDATE    = CAR2) ∨ (ERREUR(76))))
  ∨( (CAR1 = 'R')                  ∧ ((SOULIRESUME  = CAR2) ∨ (ERREUR(77))))
  ∨( (CAR1 = 'A')                  ∧ ((SOULIAUTEUR  = CAR2) ∨ (ERREUR(78))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '1')) ∧ ((SOULIENTNIV1 = CAR2) ∨ (ERREUR(79))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '2')) ∧ ((SOULIENTNIV2 = CAR2) ∨ (ERREUR(80))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '3')) ∧ ((SOULIENTNIV3 = CAR2) ∨ (ERREUR(81))))
  ∨(((CAR1 = 'E') ∧ (CAR3 = '4')) ∧ ((SOULIENTNIV4 = CAR2) ∨ (ERREUR(82))))
  ∨( (CAR1 = 'F')                  ∧ ((SOULIFIGURE  = CAR2) ∨ (ERREUR(83))))
  ∨( (CAR1 = 'L')                  ∧ ((SOULILISTE   = CAR2) ∨ (ERREUR(84))))
  ∨( (CAR1 = 'P')                  ∧ ((SOULIPARA    = CAR2) ∨ (ERREUR(85))))
  ∨( (CAR1 = 'N')                  ∧ ((SOULINOTE    = CAR2) ∨ (ERREUR(86))))
  ∨(((CAR1 = 'T') ∧ (CAR3 = 'A')) ∧ ((SOULITABLE   = CAR2) ∨ (ERREUR(87))))
  ∨( (CAR1 <> 'T','D','R','A','E','F','L','N') ∧ (ERREUR(88))
   )
 )


)
```

- Procédure appelante :

  MODELEMBASE

- Procédures appelées :

  LECDEUXPOINT
  LECTURECARACTERE
  DECODAGEVALEUR
  LECSLASH
  ERREUR

- Algorithme logique :

  début
    lecture des caractères jusqu'au ':' dans le fichier source ;
    lecture des caractères ' ' dans le fichier source ;
    lecture d'un caractère dans le fichier source (permet la
    reconnaissance du type d'élement de structure) ;
    lecture des caractères jusqu'au ':' ou jusqu'au '.' dans le
    fichier source suivant le cas ;
    lecture des caractères ' ' dans le fichier source ;
    lecture des caractères déterminant la nouvelle valeur à
    assigner à la variable ;
    assignation de la nouvelle valeur à la variable spécifiée
    par le type d'élément de structure et la variable C1 ;
    lecture des caractères jusqu'au '\' dans le fichier source
  fin

- Spécification des variables :

  CH   : caractère lu dans le fichier source
  CH1   : caractère lu dans le fichier source
  N3   : variable entière résultant du décodage
  FIN   : booléen indiquant la fin du fichier source
  BOOL : booléen utilisé dans le cas du centrage et/ou de la justification.
          Sa valeur sera 'true' si on a un cas de centrage, sinon 'false'
  TYSOU : variable de mémorisation d'un type de soulignement
  TYFON : variable de mémorisation d'un type de fonte
  NIV   : niveau d'une entte
  PAS   : indicateur d'erreur

1.2.4. PROCEDURE PROMARGE

- Entrée :

rien

- Sortie :

rien

- Effet :

Cette procédure analyse les caractères contenus dans le fichier source, afin de déterminer la variable globale pré-définie correspondant à la marge que l'on désire modifier.

- Pré-condition :

rien

- Post-conditions :

La procédure utilise des caractères du fichier source nécessaires pour la détermination de la variable globale à redéfinir.
CAR1 : détermine le type d'élément de structure ,
CAR3 : détermine la valeur à assigner à la variable.

```
(
 (
  ((C1 = 'G')
  ∧((((CAR1 = 'T') ∧(CAR2 = 'I')) ∧((MARGEGAUCHETITRE   = CAR3)∨(ERREUR(90))))
   ∨( (CAR1 = 'D')                 ∧((MARGEGAUCHEDATE    = CAR3)∨(ERREUR(91))))
   ∨( (CAR1 = 'A')                 ∧((MARGEGAUCHEAUTEUR  = CAR3)∨(ERREUR(92))))
   ∨( (CAR1 = 'R')                 ∧((MARGEGAUCHERESUME  = CAR3)∨(ERREUR(93))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '1')) ∧((MARGEGAUCHEENTNIV1 = CAR3)∨(ERREUR(94))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '2')) ∧((MARGEGAUCHEENTNIV2 = CAR3)∨(ERREUR(95))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '3')) ∧((MARGEGAUCHEENTNIV3 = CAR3)∨(ERREUR(96))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '4')) ∧((MARGEGAUCHEENTNIV4 = CAR3)∨(ERREUR(97))))
   ∨( (CAR1 = 'F')                 ∧((MARGEGAUCHEFIGURE  = CAR3)∨(ERREUR(98))))
   ∨( (CAR1 = 'L')                 ∧((MARGEGAUCHELISTE   = CAR3)∨(ERREUR(99))))
   ∨( (CAR1 = 'P')                 ∧((MARGEGAUCHEPARA    = CAR3)∨(ERREUR(100))))
   ∨( (CAR1 = 'N')                 ∧((MARGEGAUCHENOTE    = CAR3)∨(ERREUR(101))))
   ∨(((CAR1 = 'T') ∧(CAR2 = 'A')) ∧((MARGEGAUCHETABLE   = CAR3)∨(ERREUR(130))))
   ∨( (CAR1 <> T,D,A,R,E,F,L,P,N ) ∧(ERREUR(128)))
  )
  ∨
  ((C1 = 'D')
  ∧((((CAR1 = 'T') ∧(CAR2 = 'I')) ∧((MARGEDROITETITRE   = CAR3)∨(ERREUR(102))))
   ∨( (CAR1 = 'D')                 ∧((MARGEDROITEDATE    = CAR3)∨(ERREUR(103))))
   ∨( (CAR1 = 'A')                 ∧((MARGEDROITEAUTEUR  = CAR3)∨(ERREUR(104))))
   ∨( (CAR1 = 'R')                 ∧((MARGEDROITERESUME  = CAR3)∨(ERREUR(105))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '1')) ∧((MARGEDROITEENTNIV1 = CAR3)∨(ERREUR(106))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '2')) ∧((MARGEDROITEENTNIV2 = CAR3)∨(ERREUR(107))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '3')) ∧((MARGEDROITEENTNIV3 = CAR3)∨(ERREUR(108))))
   ∨(((CAR1 = 'E') ∧(CAR2 = '4')) ∧((MARGEDROITEENTNIV4 = CAR3)∨(ERREUR(109))))
   ∨( (CAR1 = 'F')                 ∧((MARGEDROITEFIGURE  = CAR3)∨(ERREUR(110))))
   ∨( (CAR1 = 'L')                 ∧((MARGEDROITELISTE   = CAR3)∨(ERREUR(111))))
   ∨( (CAR1 = 'P')                 ∧((MARGEDROITEPARA    = CAR3)∨(ERREUR(112))))
   ∨( (CAR1 = 'N')                 ∧((MARGEDROITENOTE    = CAR3)∨(ERREUR(113))))
   ∨(((CAR1 = 'T') ∧(CAR2 = 'A')) ∧((MARGEDROITETABLE   = CAR3)∨(ERREUR(131))))
   ∨( (CAR1 <> T,D,A,R,E,F,L,P,N) ∧(ERREUR(129)))
  )
  ∨
  ((C1 = 'S') ∧((MARGESUPDOCU = CAR3) ∨(ERREUR(132))))
  ∨
  ((C1 = 'I') ∧((MARGEINFDOCU = CAR3) ∨(ERREUR(133))))
 )
 ∧
  (((ERREUR (114)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHETITRE   - MARGEDROITETITRE   < 0))
  ∨((ERREUR (115)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEDATE    - MARGEDROITEDATE    < 0))
  ∨((ERREUR (116)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHERESUME  - MARGEDROITERESUME  < 0))
  ∨((ERREUR(117)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEAUTEUR  - MARGEDROITEAUTEUR  < 0))
  ∨((ERREUR(118)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEENTNIV1 - MARGEGAUCHEENTNIV2 < 0))
  ∨((ERREUR(119)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEENTNIV2 - MARGEDROITEENTNIV2 < 0))
  ∨((ERREUR(120)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEENTNIV3 - MARGEDROITEENTNIV3 < 0))
  ∨((ERREUR(121)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEENTNIV4 - MARGEGAUCHEENTNIV4 < 0))
  ∨((ERREUR(122)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEFIGURE  - MARGEDROITEFIGURE  < 0))
  ∨((ERREUR(123)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHELISTE   - MARGEDROITELISTE   < 0))
  ∨((ERREUR(124)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHEPARA    - MARGEDROITEPARA    < 0))
  ∨((ERREUR(125)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHENOTE    - MARGEDROITENOTE    < 0))
  ∨((ERREUR(126)
   ∧(LARGEURFEUILLEDOCU - MARGEGAUCHETABLE   - MARGEDROITETABLE   < 0))
  ∨((ERREUR(127)
   ∧(LONGUEURFEUILLEDOCU - MARGEINFDOCU      - MARGESUPDOCU       < 0))
  )
  ∨
  ((ERREUR(1)) ∨ (ERREUR (128))
 )
```

- Procédure appelante :

MODELEMBASE

- Procédures appelées :

LECTURECARACTERE
LECDEUXPOINT
LECBLANC
LECSLASH
DECODAGEVALEUR
ERREUR

- Algorithme logique :

```
début
  lecture des caractères jusqu'au ':' du fichier source ;
  lecture des caractères ' ' du fichier source ;
  lecture d'un caractère du fichier source (a);
  lecture des caractères jusqu'au ':' du fichier source ;
  lecture des caractères ' ' du fichier source ;
  lecture d'un caractère du fichier source (b);
  lecture des caractères jusqu'au ':' du fichier source ;
  lecture des caractères ' ' du fichier source ;
  decodage de la valeur de la marge ;
  suivant le caractère (a) - désignant le type de marge -
  et le caractère (b) - désignant le type de l'élément
  de structure -  assignation de la nouvelle valeur à la
  variable de la marge choisie ;
  lecture des caractères jusqu'au slash ;
fin
```

- Spécification des variables :

CH2, CH1, CH : caractère lu dans le fichier source
TYPEM : caractère déterminant le type de marge
TYPEE : caractère déterminant le type de l'élément de structure
VALEUR : variable entière indiquant la nouvelle valeur en nombre de millimètres de la marge à modifier
FIN : booléen indiquant la fin du fichier source

NIV : niveau d'une entête

1.2.5. PROCEDURE PRODIVERS

- Entrees :

C1 : premier caractere lu dans le fichier source
C2 : deuxieme caractere lu dans le fichier source

- Sortie :

rien

- Effet :

Cette procédure analyse les caractères contenus dans le fichier source, afin de déterminer la variable globale pré-définie que l'on désire modifier.
Prodivers traite le cas du décalage de la première ligne du paragraphe, de l'espace entre entités , de l'initialisation du numéro des notes et des figures, du type d'implémentation des listes, du nombre de niveau à indiquer dans la table des matières.

- Pré-conditions :

$(C1 \wedge C2 \quad \wedge (C1 = D, I, N, T \text{ ou } E))$

- Post-conditions :

La procédure utilise des caractères du fichier source nécessaires à la détermination de la variable globale à modifier.
CAR1 : détermine le type d'élément de structure
CAR2 : détermine la valeur à assigner à la variable

```
(
 (
  ((C1 = 'D') ∧
  (((CAR1 = 'F') ∧((DECAPARA = CAR2) ∨(ERREUR(134))))
  ∨((CAR1 = 'L') ∧ ((DECALISTE = CAR2)∨(ERREUR(135))))
  ∨((CAR1 <> 'F','L') ∧ (ERREUR(136)))))

 ∨((C1 = 'E') ∧ ((INTERENTITE = CAR2) ∨ (ERREUR(137))))

 ∨((C1 = 'N') ∧
   (((CAR1 = 'N') ∧((NUMERONOTE  = CAR2) ∨ (ERREUR(138))))
  ∨((CAR1 = 'F') ∧((NUMEROFIG   = CAR2) ∨(ERREUR(139))))
   ∨((CAR1 = 'P') ∧((NUMEROPAGE = CAR2 - 1 ) ∨ (ERREUR(132))))
   ∨((CAR1 = 'E') ∧ ((NUMEROENTETE = CAR2 -1)∨ (ERREUR(133))))
   ∨((CAR1 <> 'N','F','P','E') ∧(erreur(140)))))

 ∨((C1 = 'I') ∧ ((IMPLEMLISTE = CAR2) ∨ (ERREUR(141))))

 ∨((C1 = 'T') ∧ ((TABLENIV    = CAR2) ∨ (ERREUR(142))))

 ∨(c1 = 'D','I','N','T' OU 'E') ∧  (erreur(143))
 )
∨
 (ERREUR(1))
)
```

- Procédure appelante :

MODELEMBASE

- Procedures appelées :

LECTURECARACTERE
LECDEUXPOINT
LECBLANC
LECSLASH
DECODAGEVALEUR
ERREUR

- Algorithme logique :

```
début
  lecture des caracteres jusqu'au ':' du fichier source ;
  lecture des caracteres ' ' du fichier source ;
  lecture d'un caractere du fichier source ;
  traitement en fonction de ce caractere ;
  lecture des caracteres jusqu'au '\' ;
fin
```

- Specification des variables

```
CH     : caractère lu dans le fichier source
TYPEV  : caractère determinant le type de variable
TYPEE  : caractère determinant le type de l'élement de structure
VALEUR : variable entiere résultat du decodage
FIN    : booléen indiquant la fin du fichier source
```

1.3. PROCEDURE RECLEX

- Entrée :

PRECP : pointeur vers le record courant

- Sorties :

CH : caractère en sortie (utilisé dans le cas où l'on a eu un caractère spécial : format : \‹type d'accent› ‹caractère›)
CARSPEC : booléen indiquant la rencontre d'un caractère spécial
BOOLISTE : booléen indiquant la rencontre d'un séparateur de liste
PREC : pointeur vers le record créé
TROUVER : booléen indiquant que la séquence a été reconnue
FANION : booléen indiquant la rencontre d'au moins 2 boas dans le fichier source
IMPL : booléen indiquant le passage dans la procédure RECIMPL
IERRECORD : pointeur contenant la racine de l'arbre de structure

- Effet :

Cette procédure analyse une séquence lexicale et crée un record en fonction de ce qui est décodé.
Au cas où le premier caractère lu dans le fichier source est encore un boa, dans ce cas traitement de reconnaissance d'implémentation ;
A la sortie de la procédure, on est positionné sur le dernier caractère de la séquence analysée.

- Pré-condition :

(
PRECP
)

- Post-conditions :

La procédure utilise les caractères du fichier source afin de déterminer les variables à modifier.
CAR1 : détermine le traitement à effectuer ,
CAR2 : détermine le traitement à effectuer dans le cas où CAR1 n'est pas suffisant ,
CAR3 : détermine la valeur décodée à assigner à la variable à modifier

(
((CAR1 = '\')
∧(FANION = FALSE)
∧(reconnaissance d'implémentation effectuée)
∧(IMPL = TRUE)
∧(BOOLISTE = NEWLISTE)
∧(TROUVER)
)

∨((CAR1 = ' ',' ',' ',',',';')
∧(((CAR2 = 'a','e','i','o','u','\') ∧(CARSPEC = TRUE)
∧(CH a une valeur)
∧(TROUVE = TRUE))
∨((CARSPEC =FALSE),∧(CH n a pas de valeur)
∧(TROUVER = FALSE)))
)
)

∨(((CAR1 = E) ∧ (record de type entete crée) ∧(TROUVER = TRUE))
∨((CAR1 = P) ∧ (record de type paragraphe crée) ∧(TROUVER = TRUE))
∨((CAR1 = L) ∧ (record de type liste crée) ∧(TROUVER = TRUE))
∨((CAR1 = C) ∧ (record de type chapitre crée) ∧(TROUVER = TRUE))
∨((CAR1 = F) ∧ (record de type figure crée) ∧(TROUVER = TRUE))
∨((CAR1 = U) ∧ (record de type up crée) ∧(TROUVER = TRUE))
∨((CAR1 = D) ∧ (record de type down crée) ∧(TROUVER = TRUE))
∨((CAR1 = T) ∧ (record de type titre crée) ∧(TROUVER = TRUE))
∨((CAR1 = A) ∧ (record de type auteur crée) ∧(TROUVER = TRUE))
∨((CAR1 = R) ∧ (record de type resume crée) ∧(TROUVER = TRUE))
∨((CAR1 = N) ∧ (record de type note crée) ∧(TROUVER = TRUE))
)

∨((CAR1 <> ' ',",\,',',,,E,P,L,C,F,U,T,A,R,N,D)∧(TROUVER = FALSE)
∧(OK = TRUE))

∨(ERREUR(1))
)

- Procédure appelante :

ARBSTRUC

- Procédures appelées :

LECPOINT
CREERRECORD
RECHERCHERDIMENTION
LECTURECARACTERE
RECHERCHERNIVEAU
RECIMPL

- Algorithme logique :

```
début
  lecture d'un caractère dans le fichier source ;
  si le caractère lu est
    - un boa alors reconnaissance d'implémentation ;
    - un accent alors décodage du caractère et renvoi de son numéro ASCII
    - autre chose , alors création d'un record dont le type est défini par
      le caractère lu et liaison avec les records de l'arbre de
      structure déjà créé.
fin
```

- Spécification des variables

TEST : ensemble de deux caractères lus dans le fichier source
SORTE : type de record à créer
TROUV : booléen indiquant la découverte d'un caractère spécial
PASTROUVE : booléen indiquant l'échec lors de la recherche d'un caractère spécial
NEWLISTE : booléeh indiquant la rencontre de trois boas consécutifs
IND1 : variable contenant le niveau d'un chapitre ou la largeur d'une figure
IND2 : variable contenant la hauteur d'une figure
NUMERO : variable contenant la valeur ASCII à assigner à CH
MEMREC : pointeur prenant la valeur initiale de PRECP lors de la création d'un record

1.3.1. PROCEDURE CREERRECORD

- Entrees :

PRECP : pointeur vers le dernier record crée de l'arbre de structure
IND1 : dans le cas où le type du record à créer est un chapitre, ind1 est son niveau ;
dans le cas où le type du record à créer est une figure, ind1 est sa hauteur ;
dans les autres cas, ind1 n'a pas de signification
IND2 : dans le cas où le type du record à créer est une figure, ind2 est sa largeur ;
dans les autres cas, ind2 n'a pas de signification
SORTE : type de record à créer

- Sorties :

CREER : booléen indiquant si un record a été crée
PREC : pointeur vers le record crée
IERRECORD : pointeur vers le premier record de l'arbre, une valeur ne sera assignée que si l'on a créé un record de type document

- Effet :

Cette procédure crée un record, son type est spécifié par la variable SORTE. Suivant son type un certain nombre de valeurs sont assignées aux éléments du record.
Le record crée sera relié au record précédent de l'arbre de structure.

- Pré-conditions :

(
((SORTE = chapitre)∧ IND1)) ∨((SORTE = figure) ∧ IND1 ∧ IND2))
∧(PRECP)
∧(CREER = false)
)

- Post-conditions :

(
∧(record PREC crée)
∧(CREER = true)
∧((SORTE = document) IERRECORD)
∧(PREC'.TYPE = SORTE)
∧(lien effectué entre PREC et l'élément précédent de l'arbre)
∧(éléments du record remplis - cfr listings)
)

- Procédures appelantes :

ARBRESTRUC
RECLEX

- Procédure appelée :

LIEN

- Algorithme logique :

debut
création d'un record ;
suivant la valeur de la variable SORTE, remplissage du record crée ;
liaison du record crée au dernier record de l'arbre de structure ;
fin

- Spécification des variables :

rien

1.3.2. PROCEDURE LIEN

- Entrées :

PRECP : pointeur vers le dernier record de l'arbre de structure
PREC : pointeur vers le record à relier

- Sortie :

rien

- Effet :

Cette procedure effectue le lien entre le dernier record de l'arbre de structure et le record à ajouter

- Pré-conditions :

(
PREC ∧ PRECP
)

- Post-conditions :

```
(
  ((PREC^.TYPEUILLE = TITRE      ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = AUTEUR     ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = DATE       ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = RESUME     ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = ENTETE     ) ∧(PRECP^.POINTSUCCESS1 = PREC))
 ∨((PREC^.TYPEUILLE = PARAGRAPHE ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = FIGURE     ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = LISTE      ) ∧(PRECP^.POINTSUIVANT  = PREC))
 ∨((PREC^.TYPEUILLE = NOTEBASPAGE) ∧
     ( ((PRECP^.POINTSUCCESS1 = NIL) ∧ (PRECP^.POINTSUCCESS1 = PREC))
       ∨((PRECP^.POINTSUCCESS1 <> NIL) ∧ (Lien du dernier élement
          de la liste des records de PRECP^.POINTSUCCESS1 et PREC))
 ∨((PREC^.TYPEUILLE = CHAPITRE) ∧
     (  ((PREC^.NIVEAUCHAPITRE = 1)
       ∧( ((Il existe au moins un chapitre de niveau 1 dans l'arbre)
           ∧(dernier chapitre de niveau 1 de l'arbre ^.POINTSUIVANT = PREC))
         ∨( (Il n'existe pas un chapitre de niveau 1 dans l'arbre)
           ∧(PRECP^.POINTSUIVANT = PREC))
         )
       )

    ∨(  (PREC^.NIVEAUCHAPITE = 2)
      ∧( ( (Il existe au moins un chapitre de niveau 2 dans l'arbre et un
             chapitre de niveau 1 n'a pas été ajouté depuis
             la création du dernier chapitre de niveau 2)
           ∧(dernier chapitre de niveau 2 ^.POINTSUIVANT = PREC))
         ∨( (Il n'existe pas de chapitre de niveau 2 dans l'arbre)
           ∧(dernier chapitre de niveau 1 ^.POINTSUCCESS2 = PREC))
         ∨(ERREUR(200))
         )
       )

    ∨( (PREC^.NIVEAUCHAPITRE = 3)
      ∧(  ( (Il existe au moins un chapitre de niveau 3 dans l'arbre et un
             chapitre de niveau 1 ou 2 n'a pas été rencontré depuis la
             création du dernier chapitre de niveau 3)
           ∧(dernier chapitre de niveau 3 ^.POINTSUIVANT = PREC))
         ∨( (Il n'existe pas de chapitre de niveau 3 dans l'arbre)
           ∧(le dernier chapitre rencontré est de niveau 2)
           ∧(dernier chapitre de niveau 2 ^.POINTSUCCESS2 = PREC))
         ∨(ERREUR(201))
         )
       )
     ∨
      ( (PREC^.NIVEAUCHAPITRE = 4)
       ∧(  ( (il existe au moins un chapitre de niveau 4 dans l'arbre et un chapitre
              de niveau 1, 2 ou 3 n'a pas été ajouté depuis la rencontre
              avec le précédent chapitre de niveau 4)
            ∧(dernier chapitre de niveau 4 ^.POINTSUIVANT = PREC))
          ∨( (Il n'existe pas de chapitre de niveau 4 dans l'arbre)
            ∧(le dernier chapitre rencontré est de niveau 3)
            ∧(dernier chapitre de niveau 3 ^.POINSUCCESS2 = PREC))
          ∨(ERREUR(202))
          )
       )
   )
)
```

- Procédure appelante :

CREERRECsORD

- Procedures appelées :

FINNOTEBASPAGE
ERREUR

- Algorithme logique :

```
debut
  suivant le type du dernier record de l'arbre de structure et du record
  à ajouter : lien
fin
```

- Spécification des variables :

```
MEM       : pointeur vers un record de l'arbre de structure
PRECMEM   : pointeur vers un record de l'arbre de structure (lors du
            parcours de l'arbre afin de trouver le dernier record
            qui n'est pas une note bas de page, à la fin du parcours,
            ce pointeur contient la valeur souhaitée)
I         : compteur
TABLEAU1  : pointeur vers un record de type chapitre de niveau 1
TABLEAU2  : pointeur vers un record de type chapitre de niveau 2
TABLEAU3  : pointeur vers un record de type chapitre de niveau 3
TABLEAU4  : pointeur vers un record de type chapitre de niveau 4
```

1.3.3. PROCEDURE RECIMPL

- Entree :

PREC : pointeur vers le record courant de l'arbre de structure

- Sorties :

TROUVER : booléen indiquant qu'une séquence a été reconnue
LISTE : booléen indiquant qu'un séparateur de liste a été reconnu

- Effet :

Cette procédure a comme but de modifier un type d'implémentation typographique pré-défini. Ce type correspond à une séquence contenue dans le fichier source (caractérisée par un double bœas) Lors de la rencontre d'un triple bœas dans le fichier source, un indicateur booléen (LISTE) est initialisé à 'true' (sinon sa valeur est 'false') A la fin de la procédure, le dernier caractère lu a été celui de la fin de la séquence.

- Pré-condition :

rien

- Post-conditions :

La procédure utilise des caractères lus dans le fichier source,
CAR1 : caractère déterminant la variable à modifier,
CAR2 : caractère déterminant la variable à modifier (au cas où CAR1 ne suffit pas),
CAR3 : est la valeur que doit prendre la variable modifiée

```
(
  ( (CAR1 = \) ∧ (LISTE =TRUE))

∀(((CAR1 = C) ∧ (CAR2 = R)) ∧ ((TROUVER = TRUE) ∧ (PREC^.SAUTLIGNE = TRUE)))
∀(((CAR1 = C) ∧ (CAR2 = E)) ∧ ((TROUVER = TRUE) ∧ (PREC^.CENTRAGE = TRUE)))
∀(((CAR1 = C) ∧ (CAR2 <> R,E)) ∧ ((TROUVER =FALSE)∧(ERREUR(203))))

∀( (CAR1 = J) ∧ ((TROUVER = TRUE)∧(PREC^.SAUTLIGNE = FALSE)))

∀(((CAR1 = P) ∧ (CAR2 = A)) ∧ ((TROUVER = TRUE ) ∧ (PREC^.SAUTPAGE = TRUE)))
∀(((CAR1 = P) ∧ (CAR2 = O)) ∧( ((TROUVER = TRUE ) ∧ (PREC^.POLICE = CAR3)))
                              ∨((TROUVER = FALSE) ∧ (ERREUR(205)))))
∀(((CAR1 = P) ∧ (CAR2 <> A,O))∧((TROUVER = FALSE) ∧ (ERREUR(206))))

∀(((CAR1 = F) ∧ (CAR2 = F)) ∧ ((TROUVER = TRUE ) ∧ (PREC^.SAUTPAGE = TRUE)))
∀(((CAR1 = F) ∧ (CAR2 = O)) ∧( ((TROUVER = TRUE ) ∧ (PREC^.FONTE = CAR3))
                              ∨((TROUVER = FALSE) ∧ (ERREUR(207)))))
∀(((CAR1 = F) ∧ (CAR2 = I)) ∧( ((TROUVER = TRUE) ∧
                                ( (PREC^.SOULIGNEMENT = PREC^.SOUAN)
                                ∧(PREC^.FONTE = PREC^.FONAN)
                                ∧(PREC^.POLICE = PREC^.POLAN) ) )
                              ∨((TROUVER = FALSE) (ERREUR(208)))))
∀(((CAR1 = F) ∧ (CAR2 <> F,I,O )) ∧((TROUVER = FALSE) ∧ (ERREUR(209))))

∀(((CAR1 = L) ∧ (CAR2 = $)) ∧ ((TROUVER = TRUE) ∧ (PREC^.SAUTLIGNE = TRUE)))
∀(((CAR1 = L) ∧ (CAR2 = S)) ∧( ((TROUVER = TRUE) ∧ (PREC^.LISTE = CAR3))
                              ∨((TROUVER = FALSE) ∧ (ERREUR(210)))))
∀(((CAR1 = L) ∧(CAR2 <> S,$)) ∧ ((TROUVER = FALSE) ∧ (ERREUR(211))))

∀( (CAR1 = S) ∧( ((TROUVER = TRUE ) ∧ (PREC^.SOULIGNEMENT = CAR3))
               ∨((TROUVER = FALSE) ∧ (ERREUR(212)))))

∀( (CAR1 = D) ∧( ((TROUVER = TRUE ) ∧ (DECALAGEPARAGRAPHE = CAR3))
               ∨((TROUVER = FALSE) ∧ (ERREUR(213)))))

∀(((CAR1 = M) ∧ (CAR2 = G)) ∧( ((TROUVER = TRUE ) ∧ (PREC^.MARGEGAUCHE = CAR3))
                              ∨((TROUVER = FALSE) ∧ (ERREUR(214)))))
∀(((CAR1 = M) ∧ (CAR2 = D)) ∧( ((TROUVER = TRUE ) ∧ (PREC^.MARGEDROITE = CAR4))
                              ∨((TROUVER = FALSE) ∧ (ERREUR(215)))))
∀(((CAR1 = M) ∧ (CAR2 <> G,D))∧((TROUVER = FALSE) ∧ (ERREUR(216))))
∀((CAR1 <> M,D,S,L,I,F,P,C,J )∧(ERREUR(217))))

)
```

- Procédure appelante :

RECLEX

- Procédures appelées :

LECTURECARACTERE
LECPOINT
DECODAGEVALEUR
ERREUR

- Algorithme logique :

```
debut
  lecture d'un caractère du fichier source ;
  détermination de la variable à modifier en fonction du caractère lu
fin
```

- Spécification des variables :

MEMREC	: pointeur vers un record de l'arbre de structure
TAMP	: ensemble de caractères lus dans le fichier source et servant à la détermination de la variable à modifier
NB	: variable résultat du décodage
OK	: booléen indicateur d'une variable à modifier
FIN	: booléen indiquant la fin du fichier source

1.3.4. PROCEDURE RECHERCHERNIVEAU

- Entrée :

  rien

- Sortie :

  IND1 : niveau du chapitre

- Effet :

  Cette procedure recherche dans le fichier source le niveau d'un chapitre

- Pré-condition :

  rien

- Post-conditions :

  (
  ((IND1 < 4) ∨ (ERREUR(2))) ∨ ERREUR(1)
)

- Procédure appelante :

  RECLEX

- Procédures appelées :

  LECTURECARACTERE
  LECPOINT
  LECDEUXPOINT

- Algorithme logique :

  début
  lecture des caractères jusqu'au ':' dans le fichier source;
  lecture des caractères jusqu'au ':' dans le fichier source ;
  lecture des caractères ' ' dans le fichier source ;
  lecture d'un caractère dans le fichier source ;
  transformation du caractère lu en entier ;
  test de la valeur de l'entier;
  lecture des caractères jusqu'au "." dans le fichier source
  fin

- Spécification des variables :

  CH : caractere lu dans le fichier source
  FIN : booléen indiquant la fin du fichier source

1.3.5. PROCEDURE RECHERCHERDIMENTION

- Entrée :

  rien

- Sorties :

  IND1 : hauteur de la figure
  IND2 : largeur de la figure

- Effet :

  Cette procédure recherche dans le fichier source la hauteur et la largeur d'une figure

- Pré-condition :

  rien

- Post-conditions :

  (
  (IND1 ∧ IND2) ∧ (ERREUR(1)
)

- Prodédure appelante :

  RECLEX

- Procédure appelée :

  LECTURECARACTERE
  LECPOINT

- Algorithme logique :

  début
  lecture des caracteres jusqu'à obtenir un caractère numérique ;
  lecture de tous les caractères numériques et décodage ;
  assignation de la valeur obtenue à la variable IND1 ;
  lecture des caracteres jusqu'à obtenir un caractère numérique ;
  lecture de tous les caractères numériques et décodage ;
  assignation de la valeur obtenue à la variable IND2 ;
  lecture de tous les caractères jusqu'au '.'
  fin

- Spécification des variables :

  CH : caractère lu dans le fichier source
  TROUV : booléen indiquant l'obtention de IND2
  IER : booléen indiquant l'obtention de IND1
  FIN : booléen indiquant la fin du fichier source
  NB : variable entiere resultat du decodage.

1.3.6. PROCEDURE PARCOURSFINNOTE

- Entrée :

  PREC : pointeur vers le record courant à traiter dans l'arbre

- Sortie :

  MEMREC : pointeur vers le dernier record de l'arbre qui n'est pas une note bas de page

- Effet :

  Cette procédure rend l'adresse du dernier élément de l'arbre de structure qui n'est pas du type note bas de page.

- Pré-condition :

  (
  PREC = pointeur vers un élément de l'arbre
)

- Post-condition :

  (
  MEMREC = pointeur vers le dernier élément de l'arbre qui n'est pas de type note bas de page.
)

- Procédure appelante :

  FINNOTEBASPAGE

- Procédure appelée :

  rien

- Algorithme logique :

  début
  parcours récursif des éléments de l'arbre de structure avec mémorisation de chaque record qui n'est pas une note bas de page
  fin

- Spécification des variables :

  rien

1.3.7. PROCEDURE FINNOTEBASPAGE

- Entrée :

  rien

- Sortie :

  PREC : pointeur vers le record précédant la note bas de page

- Effet :

  Cette procédure recherche l'adresse du dernier élément de l'arbre de structure qui a engendré la note bas de page et rend l'adresse du dernier mot créé de cet élément de structure (dans la variable globale MEMMOT)

- Pré-condition :

  (
  ADRIERRECORD = adresse du premier record de l'arbre de structure
)

- Post-conditions :

  (
  (MEMMOT = adresse du dernier mot du record contenant l'élément de structure qui comprend la note bas de page)
  ∧ PREC = adresse du record contenant l'élément de structure qui comprend la note bas de page)
)

- Procédures appelantes :

  RECLEX
  LIEN

- Procédure appelée :

  PARCOURSFINNOTE

- Algorithme logique :

  début
  recherche de l'adresse du record de l'élément de structure qui contient la note bas de page ;
  recherche de l'adresse du dernier mot du record de l'élément de structure qui contient la note bas de page
  fin

- Spécification des variables :

  ADRESSEMOT : variable utilisée pour rechercher le pointeur auquel on associera le mot créé

1.4. PROCEDURE REMPLIRMOT

- Entrées :

CREER : booleen indiquant qu'un élement a été crée
CH : caractère lu dans le fichier source
PREC : pointeur vers le record courant de l'arbre
PRECP : pointeur vers l'avant dernier record de l'arbre
NUMEROFIG : numéro de la figure courante
NUMERONOTE : numéro de la note bas de page courante
TABLEAU : tableau d'entier contenant le nombre de chapitres (nombre par niveau) déja rencontrés. (correspondant au numéro de l'entete)
MEMMOT : pointeur vers le dernier mot crée
DEJASEP : booléen indique qu'un separateur a déja été rencontré
IERPAS : booléen indique le passage dans la procédure pour l'élément de structure courant
IERMOT : booléen indique qu'un mot existe déja pour l'élement de structure courant
NEWLISTE : booléen indique le premier élément d'une liste

- Sorties :

Mêmes éléments que ceux en entrée mis à jour.

- Effet :

Cette procédure a comme but de remplir des «mots»

- Procédure appelante :

ARBRSTRUC

- Procéduros appelées :

NUMEROTATIONFIG
NUMEROTATIONNOTE
IMPLEMENTATIONLISTE
NUMEROTATIONENTETE

- Pre-conditions :

(
CREER ∧CH ∧ PREC ∧ PRECP ∧ NUMEROFIG ∧ NUMERONOTE ∧ TABLEAU
∧ MEMMOT ∧ DEJASEP ∧ IERPAS ∧ IERMOT ∧ NEWLISTE
)

- Post-conditions :

(
(((IERMOT = TRUE)∧(NEWLISTE = TRUE))
∧((((PREC'.TYPEUILLE = LISTE)
((mot crée contenant le type d'implementation desire)
∧(((NEWLISTE = TRUE) ∧ (PREC'.POINTMOT = motcrée))
∨((NEWLISTE = FALSE) ∧ (MEMMOT'.POINTMOT = motcrée))
∧(IERMOT = FALSE)
)
)
∨
(((PREC'.TYPEUILLE = NOTEBASPAGE)
∧ ((mot crée 1 contenant le sigle (en bas de page) et le numéro de la note)
∧(mot crée 2 contenant le sigle (dans le texte) et le numéro de la note)
∧(PREC'.POINTMOT = mot crée 1)
∧(le dernier mot de la chaine des mots associée à PRECP '.POINTMOT = mot crée 2)
∧(IERMOT = FALSE)
)
)
∨
(((PREC'.TYPEUILLE = FIGURE)
∧((mot crée 1 contenant le sigle (en dessous de la figure) et le numéro)
∧(mot crée 2 contenant le sigle (dans le texte) et le numéro)
∧(PREC'.POINTMOT = mot crée 1)
∧(le dernier mot de la chaine des mots associée à PRECP '.POINTMOT = mot crée 2)
∧(IERMOT = FALSE)
)
)
∨
(((PREC'.TYPEUILLE = ENTETE)
((mot crée contenant le numéro de l'entete)
(PREC'.POINTMOT = mot crée)
(IERMOT = FALSE)
)
)
)
)
∧
((CH est un séparateur) ∧ (DEJASEP = TRUE))
∧
(((((IERPAS = TRUE) ∧ (CH est une separateur)) ∧ (CREER = TRUE))

```
∧((PREC^.TYFEUILLE = CHAPITRE)∧ (ERREUR(8))
∧(CREER = FALSE)
∧(mot crée)
∧( ( (IERMOT = TRUE)  ∧(PREC^.POINTMOT = mot crée)
                      ∧(mot crée^.POINTMOT = NIL)
                      ∧(IERMOT = FALSE)
                      ∧(MEMMOT = mot crée)
   )
  ∨( (IERMOT = FALSE) ∧(mot crée^.POINTMOT = NIL)
                      ∧(MEMMOT^.POINTMOT = mot crée)
                      ∧(mot crée^.POINTMOT = NIL)
   )
∧( assignation des valeurs au record mot )
∧( (( CH n'est pas un séparateur ) ∧ (DEJASEP  = FALSE))
  ∨( CH est un séparateur) ∧ (DEJASEP = TRUE))
 )
∧(IERPAS = FALSE)
∧(mot crée^.STRING[0] = CHR(1))
∧(mot crée^.STRING[1] = valeur de CH décodée)
)
∨
( (NOT (( ((IERPAS = TRUE) ∧ (CH n'est pas un séparateur))
        ∧(DEJASEP = TRUE) ∧ (CREER = TRUE))))
∧(MEMMOT^.STRING[0] = le nombre de caractères du mot)
∧(MEMMOT^.STRINT[première adresse vide] = valeur de CH décodée)
 )
)
)
```

- Procédure appelante :

  ARBSTRUC

- Procédures appelées :

  NUMEROTATIONFIG
  NUMEROTATIONNOTE
  NUMEROTATIONENTETE
  IMPLEMLISTE

- Algorithme logique :

```
début
  si le record de structure ne contient pas encore de mot
    alors, si le record est du type figure, entete, liste ou note bas de page ,
    création d'un mot contenant une référence ;
  si  nouvel élément de liste, on crée un nouveau mot ;
  si le record de structure ne contient pas encore de mot,
    alors un mot est crée, et le lien est effectué avec le record
    sinon création ou non d'un mot suivant le cas, dans le cas où un
    mot est crée, il est relié au mot précédent
fin
```

- Spécification des variables :

```
SORTESEP  : ensemble contenant les caractères "séparateurs"
PTRM      : pointeur vers une élément de type mot
```

1.4.1. PROCEDURE NUMEROTATIONFIG

- Entrées :

PRECP : pointeur vers l'élément précédent de l'élément courant de l'arbre de structure
PREC : pointeur vers l'élément courant de l'arbre de structure (de type figure)
NUMEROFIG : numéro assigné à la figure précédente

- Sorties :

NUMEROFIG : numéro assigné à la figure courante
MEMMOT : pointeur vers le mot créé

- Effet :

Cette procédure crée un mot contenant le numéro de la figure et le sigle, ce mot est relié au record de type figure.
De plus, un mot est créé et ajouté à la chaine des mots du dernier record - précédent celui du type figure - ce mot comprend une référence à la figure et est relié au dernier mot du record précédent.

- Pre-conditions :

(PREC ∧ PRECP ∧ NUMEROFIG)

- Post-conditions :

(
∧(MEMMOT créé)
∧(MEMMOT comprend le numéro de la figure et le mot 'Fig.')
∧(PREC^.POINTMOT = MEMMOT)
∧(création d'un mot contenant le mot 'FIG' suivi du numéro de la figure = MOTCREE)
∧(le dernier mot de la chaine des mots de PRECP contient le pointeur vers MOTCREE)
)

- Procédure appelante :

REMPLIRMOT

- Procédure appelée :

EXPDIX

- Algorithme logique :

```
debut
  création de deux mots ;
  initialisation de la valeur de ces mots ;
  recherche du dernier mot de l'élément de structure précédent ;
  liaison d'un des mots créé avec le dernier mot du record précédent ;
  liaison du deuxième mot créé avec le record de type figure ;
  incrementation de la variable indiquant le numéro de la dernière figure
fin
```

- Spécification des variables :

MOT : pointeur de mémorisation d'un élément de type mot
RESERVE : pointeur de mémorisation d'un élément de type mot
MOTCREE : pointeur vers un élément de type mot, il s'agit du mot qui est relié à la chaine des mots du record auquel la figure est rattachée
I : compteur
NUMERO : variable entière permettant d'effectuer un calcul sur le numéro de la figure (nécessité de transformer un type integer en type caractère)
NBCAR : variable entière indiquant le nombre de caractère du numéro de la figure
VAR1 : variable entière de travail

1.4.2. PROCEDURE NUMEROTATIONNOTE

- Entrées :

PRECP : pointeur vers l'élément précédent de l'élément courant de l'arbre de structure
PREC : pointeur vers l'élément courant de l'arbre de structure (de type note bas de page)
NUMERONOTE : numéro assigné à la note précedente

- Sorties :

NUMERNOTE : numéro assigné à la note courante
MEMMOT : pointeur vers le mot créé

- Effet :

Cette procédure crée un mot contenant le numéro de la note et le sigle, ce mot est relié au record de type note.
De plus, un mot est ajouté à la chaine des mots du dernier record - précédant celui du type note - ce mot comprend une référence à la note et est relié au dernier mot du record précedent.

- Pre-conditions :

(
PREC ∧ PRECP ∧ NUMERONOTE
)

- Post-conditions :

(
∧(MEMMOT créé)
∧(MEMMOT comprend le numéro de la note et le mot 'NB.')
∧(PREC'.POINTMOT = MEMMOT)
∧(création d'un mot contenant le mot 'NB.' suivi du numéro de la note = MOTCREE)
∧(le dernier mot de la chaine des mots de PRECP contient le pointeur vers MOTCREE)
)

- Procédure appelante :

REMPLIRMOT

- Procédure appelée :

EXPDIX

- Algorithme logique :

début
création de deux mots ;
initialisation de la valeur de ces mots ;
recherche du dernier mot de l'élément de structure précédent ;
liaison d'un des mots créé avec le dernier mot du record précédent ;
liaison du deuxième mot créé avec le record de type note ;
incrementation de la variable indiquant le numéro de la dernière note
fin

- Spécification des variables :

MOT : pointeur de mémorisation d'un élement de type mot
RESERVE : pointeur de mémorisation d'un élement de type mot
MOTCREE : pointeur vers un élement de type mot, il s'agit du mot qui est relié à la chaine des mots du record auquel la note est rattachée
I : compteur;
NUMERO : variable entière permettant d'effectuer un calcul sur le numéro de la note (nécessité de transformer un type integer en type caractère)
NBCAR : variable entière indiquant le nombre de caractères du numéro de la note
VAR1 : variable entière de travail

1.4.3. PROCEDURE NUMEROTATIONENTETE

- Entrees :

PREC : pointeur vers le dernier record crée (de type entete)
PRECP : pointeur vers le record précédant le dernier record crée (de type chapitre)
TABLEAU : tableau d'entier contenant le numéro des différents chapitres précédents

- Sorties :

TABLEAU : tableau d'entier contenant le numéro des différents chapitres précédents mis à jour
MEMMOT : pointeur vers le mot crée

- Effet :

Cette procédure crée un mot qui contient la numérotation du chapitre (exemple 1.2.1.).
La procedure effectue la mise à jour du tableau contenant le numéro des chapitres .
Le mot crée est relié au record de type entete.

- Pré-conditions :

(
TABLEAU ∧ PREC ∧ PRECP
)

- Post-conditions :

(
(MEMMOT comprend le numéro de l'entête en fonction du niveau du chapitre - dont le pointeur vers l'élement de structure est PRECP)

∧(le pointeur vers les mots de PREC est initialisé à MEMMOT)

∧((TABLEAU[PRECP^.NIVEAUCHAPITRE + I] = 0) ∧ (I > 1)
∧(PRECP^.NIVEAUCHAPITRE + I <= 6))

∧(TABLEAU[PRECP^.NIVEAUCHAPITRE] = TABLEAU[PRECP^.NIVEAUCHAPITRE] + 1)
)

- Procédure appelante :

REMPLIRMOT

- Procédure appelée :

rien

- Algorithme logique :

début
incrementation du tableau pour le niveau du chapitre de l'entete ;
creation d'un mot ;
assignation de la valeur du tableau au mot crée ;
mise à '0' dans le tableau des niveaux superieurs à celui du chapitre
fin

- Specification des variables :

I : compteur
D : variable entière
REC : variable entière

1.4.4. PROCEDURE IMPLEMLISTE

- Entrées :

NEWLISTE : booléen indiquant le premier élement d'une liste
PREC : pointeur vers le record de l'élément de structure courant (celui créé pour la liste)

- Sortie :

MEMMOT : pointeur vers le mot créé

- Effet :

Cette procédure crée un mot dont la forme est déterminée par l'implementation désirée pour la liste.
Le mot créé est relié soit au record de type liste dans le cas où NEWLISTE = 'true' , soit au mot précédent de la liste ; dans ce dernier cas, l'indicateur de saut de page est à TRUE.

- Pré-conditions :

(
NEWLISTE ∧ PREC
)

- Post-conditions :

(
((NEWLISTE = TRUE) ∧ (MEMMOT comprend l'implémentation de la liste)
∧ (PREC^.POINTMOT = ^MEMMOT)
∧ (MEMMOT^.SAUTLIGNE = TRUE)
)
∨
((NEWLISTE = FALSE) ∧ (MEMOT comprend l'implementation de la liste)
∧ (le dernier mot de la chaine associé à
PREC comprend le pointeur vers MEMMOT)
)
)

- Procédure appelante :

REMPLIRMOT

- Procédure appelée :

rien

- Algorithme logique :

```
début
  création d'un mot ;
  si premier élement de la liste alors le pointeur vers les mots
    du record courant vaudra le pointeur vers le mot crée
  sinon le pointeur du dernier mot suivant le dernier
    mot créé vaudra le pointeur vers le mot crée;
  remplissage du mot crée suivant ce qui a été défini au niveau global
fin
```

- Spécification des variables :

TYPEL : type d'implémentation désiré pour la liste
I : compteur
MOTPREC : pointeur vers le dernier mot de l'élement de structure courant (ceci avant la création d'un mot au sein de la procédure)

1.0.1 PROCEDURE LECTURECARACTERE

- Entrée :

  rien

- Sorties :

  CH : caractère lu dans le fichier source
  FIN : booléen indiquant la fin du fichier source

- Effet :

  Cette procédure lit un caractère dans le fichier source en laissant tomber les caractères de fin de ligne

- Pré-condition :

  rien

- Post-conditions :

  (
  FIN ∧ CH
)

- Procédures appelantes :

LECDEUXPOINT	LECJUSTBLANC	LECPOINT
DECODAGEVALEUR	LECSLAH	ARBSTRUC
MODELEMBASE	PROMODREDUIT	PROMODAUTRE
PROMARGE	PRODIVERS	RECLEX
RECIMPL	RECHERCHERDIMENSION	RECHERCHERNIVEAU

- Procédure appelée :

  rien

- Algorithme logique :

```
début
  si fin de fichier alors indicateur de fin à "true"
  sinon
    début
      si fin de ligne alors lecture d'un caractère ;
      lecture d'un caractère
    fin
fin
```

- Spécification des variables :

  rien

1.0.2 PROCEDURE LECDEUXPOINT

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procédure lit tous les caractère dans le fichier source, jusque la rencontre d'un caractère ':'

- Pré-condition :

  rien

- Post-conditions :

  (
  (positionnement sur le caractère ':' dans le fichier source)
  ∨(ERREUR(1))
)

- Procédures appelantes :

PROMODREDUIT	PROMODAUTRE	RECIMPL
PROMARGE	PRODIVERS	RECLEX
PRODIVERS		

- Procédures appelées :

  LECTURECARACTERE
  ERREUR

- Algorithme logique :

```
début
  tant que caractère lu <> ':', lecture de caractères dans le fichier source
fin
```

- Spécification des variables :

  CH : caractère lu dans le fichier source
  FIN : indicateur booléen de fin de fichier

1.0.3. PROCEDURE LECJUSTBLANC

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procédure lit tous les caractères dans le fichier source, jusque la rencontre d'un caractère ' '

- Pré-condition :

  rien

- Post-conditions :

  (
  (positionnement sur le caractère ' ' dans le fichier source)
  ∨ (ERREUR(1))
)

- Procédure appelante :

  RECIMPL

- Procédures appelées :

  LECTURECARACTERE
  ERREUR

- Algorithme logique :

  début
  tant que caractère lu <> ' ', lecture de caractères dans le fichier source
  fin

- Spécification des variables :

  CH : caractère lu dans le fichier source
  FIN : indicateur booléen de fin de fichier

1.0.4 PROCEDURE LECSLASH

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procédure lit tous les caractères dans le fichier source, jusque la rencontre d'un caractère '\'

- Pré-condition :

  rien

- Post-conditions :

  (
  (positionnement sur le caractère '\' dans le fichier source)
  ∨ (ERREUR(1))
)

- Procédures appelantes :

  PROMODREDUIT
  PROMODAUTRE
  PROMARGE
  PRODIVERS

- Procédures appelées :

  LECTURECARACTERE
  ERREUR

- Algorithme logique :

  début
  tant que caractère lu <> '\', lecture de caractères dans le fichier source
  fin

- Spécification des variables :

  CH : caractère lu dans le fichier source
  FIN : indicateur booléen de fin de fichier

1.0.5 PROCEDURE LECPOINT

- Entrée :

rien

- Sortie :

rien

- Effet :

Cette procedure lit tous les caractères dans le fichier source, jusque la rencontre d'un caractère '.'

- Pre-condition :

rien

- Post-conditions :

(
(positionnement sur le caractère '.' dans le fichier source)
∨(ERREUR(1))
)

- Procédures appelantes :

FINNOTEBASPAGE
RECHERCHERNIVEAU
RECHERCHERDIMENSION
PROMODREDUIT
PROMODAUTRE
RECIMPL
PROMARGE
PRODIVERS
RECLEX

- Procédures appelées :

LECTURECARACTERE
ERREUR

- Algorithme logique :

debut
tant que caractère lu <> '.', lecture de caractères dans le fichier source
fin

- Spécification des variables :

CH : caractère lu dans le fichier source
FIN : booléen indiquant la fin du fichier

1.0.6. PROCEDURE DECODAGEVALEUR

- Entrée :

CH : caractère lu dans le fichier source

- Sortie :

NB : variable entière resultat du decodage

- Effet :

Cette procédure à partir d'un caractère sur lequel on est positionné dans le fichier source, lit tous les caractères dont le code ASCII est compris entre 48 et 57 (bornes comprises), elle effectue le décodage, afin de rendre un nombre entier.

- Pré-condition :

rien

- Post-conditions :

(
NB ∨ ERREUR(1)
)

- Procédures appelantes :

PROMODREDUIT
PROMODAUTRE
PRODIVERS
RECIMPL
PROMARGE

- Procédures appelées :

LECTURECARACTERE
ERREUR

- Algorithme logique :

début
initialisation d'une variable entière ;
lecture de caractères dans le fichier source tant que le code ASCII de ces caractères est compris entre 48 et 57, et décodage
fin

- Spécification des variables :

FIN : booléen indiquant la fin du fichier source

1.0.7 PROCEDURE EXPDIX

- Entrée :

  VALEUR : variable entière

- Sortie :

  EXPDIX : résultat de l'élévation à la puissance 10

- Effet :

  Cette procédure élève un nombre à la puissance 10

- Pré-condition :

  (
  VALEUR
)

- Post-condition :

  (
  EXPDIX
)

- Procédures appelantes :

  NUMEROTATIONFIG
  NUMEROTATIONNOTE

- Procédure appelée :

  rien

- Algorithme logique :

  début
  effectuer 10 exposant (n * 10)
  fin

- Spécification des variables :

  I : compteur de boucle
  VAL : variable contenant les résultats intermédiaires

2. PROCEDURE ARBREFORMATAGE

- Entrées :

POINTDOC : pointeur vers le document
POINTPAGE : pointeur vers la première page

- Sortie :

POINTPAGE : pointeur vers la première page

- Effet :

Cette procédure, recevant le pointeur vers la racine de l'arbre de structure construit l'arbre de formatage ; elle justifie le texte, crée la chaine des entités, traite la mise en page et renvoie le pointeur vers la première page formatée du document ; à la demande elle construit également une table de matière et/ou une numérotation

- Pré-conditions :

(
POINTDOC ∧ (POINTPAGE = NIL)
)

- Post-condition :

(
POINTPAGE <> NIL
)

- Procedure appelante :

PROGRAMME PRINCIPAL

- Procedures appelées :

TABLEAU
TRTARBREFORMATAGE
MISEENPAGE
TABLEMATIERE

- Algorithme logique :

```
debut
  création des tableaux de caractères ;
  création de l'arbre de formatage ;
  miseenpage des entités ;
  si table des matieres demandée, création et gestion
      de la table des matières ;
  si pagination demandée, numérotation des pages
fin
```

- Spécification des variables :

ARBFOR	: booléen indiquant si c'est le premier élement de l'arbre de structure
COLONNAGETYPO	: booléen indiquant que la mise en page est en simple ou double colonne
INDEXTYPO	: booléen indiquant la création d'un index
TABLEMATIERETYPO	: booléen indiquant la création d'une table des matières
POINTENTITE	: pointeur vers une entité
MEMENTITE	: pointeur vers la première entité du document
ADRPAGE	: pointeur vers la page courante
MEMPAGE	: pointeur de mémorisation d'une page
IERTAB	: pointeur vers la première page constituant la table des matières
TABLEAU11	: tableau de caractères correspondant à la police numéro 10/fonte normal
TABLEAU12	: tableau de caractères correspondant à la police numéro 11/fonte normal
ECARTMOYEN	: écart moyen entre entités

2.1 PROCEDURE TABLEAU

- Entrée :

rien

- Sortie :

rien

- Effet :

Cette procédure, à partir d'un fichier d'un jeu de caractères (dont une description est donnée dans l'analyse organique), construit un tableau de dimensions des caractères. Ce tableau reprend la largeur, la hauteur au-dessus et en-dessous de la "base line" et cela pour les 128 caractères ASCII ; les dimensions sont exprimées en points

- Pré-conditions :

(
fichiers: tab 10, tab11....
)

- Post-condition :

(
TABLEAU construit
)

- Procédure appelante :

ARBREFORMATAGE

- Procédure appelée :

BINAIREDECIMALE

- Algorithme logique :

```
début
  ouverture du fichier ;
  tant que pas fin de fichier
    début
      tant que pas fin de ligne
        début
          lecture d'un caractère ;
          si positionné à l'endroit de stockage
            début
              placement du caractère dans un byte ;
              si byte est rempli
                début
                  decodage binaire/decimal du byte ;
                  si place > 127
                    début
                      incrémentation de l'indice indiquant le
                          type de dimension ;
                      réinitialisation de la place
                    fin
                fin
            fin
        fin
      lecture du caractère suivant
    fin
fin
```

- Spécification des variables :

VAL	:	variable de mémorisation du caractère
COMPTEUR	:	endroit de positionnement de la lecture du fichier
POSITION	:	position courante dans le byte
INDICE1	:	indice indiquant le type de dimension
PLACE	:	numéro ASCII du caractère
VALEUR	:	valeur décimale du caractère lu
CARA	:	caractère lu
LARGEUR	:	variable de type byte contenant une dimension encodée en binaire

2.2 PROCEDURE TRTARBREFORMATAGE

- Entrées :

  POINTDOCUMENT : pointeur vers la racine de l'arbre de stucture
  LARPOIPA : largeur d'une feuille du document

- Sorties :

  POINTENTITE : pointeur vers l'entité courante
  MEMENTITE : pointeur vers la première entité
  ARBPOR : booléen indiquant si c'est le premier record de l'arbre de structure
  COLONNAGE : booléen indiquant si mise en page en simple ou double colonne

- Effet :

  Cette procédure parcourt récursivement l'arbre de structure. Pour chaque élément de stucture, elle lui associe une entité qui contient toutes les indications typographiques de formatage de l'élément de structure ; elle chaine ces entités de manière à construire un arbre

- Pré-conditions :

  (
  (ARBPOR = TRUE) ∧
  (MEMENTITE = NIL) ∧(PDOCUMENT <> NIL) ∧(POINTENTITE = NIL)
)

- Post-conditions :

  (
  ((MEMENTITE <> NIL) ∧ (POINTENTITE <> NIL) ∧(ARBPOR = FALSE)) ∧
  ((corespondance élément de structure/entité) ∧
  (POINTMOT <> nil) ∧
  (toutes les entités sont chainées)) ∧
  (chaque entité contient toutes les indications de formatage)
)

- Procédure appelante :

  ARBREFORMATAGE

- Procédure appelée :

  TRAITEMENTCLASSIQUE

- Algorithme logique :

```
début
si il existe un record de structure
 début
   si il existe des mots pour cet élément
    début
      détermination des variables du record de l'arbre de formatage
           à partir des variables du record de l'arbre de structure
           correspondant ;
      si l'élément de structure <> chapitre, gestion du
         formatage de l'entité correspondante ;
      détermination d'autres indications de formatage propres
         à certains types d'éléments de structure ;
      si l'élément de structure est une figure
        début
          création d'une entité figure ;
          gestion des éléments de l'entité créée
        fin
      gestion de la chaine des entité
    fin
   même traitement pour l'élément de structure success1 ;
   même traitement pour l'élément de structure success2 ;
   même traitement pour l'élément de structure suivant
 fin
fin
```

- Spécification des variables :

  MARGEGAUCHETYPO : marge gauche de l'entité
  MARGEDROITETYPO : marge droite de l'entité
  INTERLIGNETYPO : espace entre lignes au sein de l'entité
  NBPOINTENTITE : largeur en point typographique de l'entité
  HAUTEURTYPO : hauteur de l'entité
  LARGEURTYPO : largeur de l'entité
  DECALAGETYPO : espace que représente le décalage, par rapport à la marge droite de l'entité, de la première ligne d'un paragraphe
  CENTRAGETYPO : booléen indiquant si l'élément typographique est centré
  FEUILLETYPO : booléen indiquant si l'entité est une feuille
  IERELIGNETYPO : booléen indiquant si c'est la première ligne de l'entité
  FONTETYPO : entier qui représente type de fonte
  SOULIGNEMENTTYPO : indique le type de soulignement de l'élément typographique
  TYPEFEUILLETYPO : indique le type d'entité
  POLICETYPO : entier qui représente le type de police
  VALEURENTITE : pointeur vers l'entité courante
  FIGENTITE : pointeur vers une entité de type figure

2.2.1. PROCEDURE TRAITEMENTCLASSIQUE

- Entrees :

POINTMOT	: pointeur vers le premier mot de l'arbre de structure
NBPOINTENTITE	: largeur en nombre de points de l'entite
MARGEGAUCHETYPO	: marge gauche de l'entite
MARGEDROITETYPO	: marge droite de l'entite
POLICETYPO	: type de police
INTERLIGNETYPO	: espace entre ligne
DECALAGE	: espace entre la premiere ligne de l'entite et la marge gauche
IERLIGNE	: pointeur vers la premiere ligne
CENTRAGETYPO	: booleen indiquant si centrage
FONTETYPO	: type de fonte
SOULIGNEMENTTYPO	: type de soulignement

- Sortie :

POINTENT : pointeur vers l'entite creee

- Effet :

Cette procedure, a partir du pointeur vers le premier mot de l'arbre de structure et des indications d'implementation typographique, construit une entite pour le record de l'arbre de structure. Cette entite contient toutes les indications de formatage

- Pre-conditions :

(
(POINTMOT <> NIL) ∧ (tous les parametres de formatage sont determines)
)

- Post-conditions :

(
((POINTMOT <> NIL) ∧ (POINTENT <> NIL) ∧
 (creation des bouts de ligne, lignes et entite) ∧
 (mise a jour des entites, lignes, bouts de ligne))
)

- Procedure appelante :

TRTARBREFORMATAGE

- Procedures appelees :

INIT
CREERRECORD
EXAMEN
TRTMOT
MISEAJOUR
TRAITEMENTCHANGEMENT

- Algorithme logique :

```
debut
  initialisation des variables de formatage ;
  creation d'une entite, d'une ligne et d'un bout de ligne ;
  initialisation de la valeur du ' ' a priori ;
  tant qu'il reste des mots
    debut
      reinitialisation des variables (courantes et de memorisation)
          de la hauteur, largeur des lignes et des bouts de ligne ;
      initialisation de la position du mot dans un buffer ;
      si c'est la premiere ligne, gestion du decalage ;
      traitement du mot ;
      mise a jour de la hauteur et largeur des lignes ;
      gestion du changement au sein des boutlignes ;
      gestion du completage ;
      mise a jour de la chaine des lignes et des entites ;
      gestion du completage ;
      gestion du centrage ;
      gestion de l'entite creee ;
    fin
fin
```

- Specification des variables :

NBCAR	: nombre de caracteres d'un mot
HAUTEURMOT	: hauteur d'un mot
LARGEURMOT	: largeur d'un mot
COMPTEURBLANC	: nombre de ' ' au sein de l'entite
COMPTEURBLANCBOUTLI	: nombre de ' ' dans un bout de ligne
COMPTEURBLANCLI	: nombre de ' ' dans une ligne
HAUTEURLI	: hauteur d'une ligne
HAUTEURBOUTLI	: hauteur d'un bout de ligne
LARGEURLI	: largeur d'une ligne
LARGEURBOUTLI	: largeur d'un bout de ligne
MEMLARGEURLI	: variable de memorisation de la largeur d'une ligne
MEMLARGEURBOUTLI	: variable de memorisation de la hauteur du bout de ligne courant
LARGEURBLANC	: largeur optimale du caractere ' '
MEMHAUTEURBOUTLI	: variable de memorisation de la hauteur du bout de ligne courant
NBLI	: nombre de ligne(s) de l'entite courante
HAUTEURENT	: hauteur de l'entite
LARGEURLISTANTARD	: largeur d'une ligne standard fixee a priori
RESTE	: variable de memorisation de la largeur de entite courante
HAUT	: hauteur d'un caractere
LARG	: largeur d'un caractere
LARGBLANC	: largeur du caractere ' ' fixee a priori
POSDEBMOT	: position d'un mot dans un buffer
NBBOUTLI	: nombre de bouts de ligne
ADRESSEBOUTLI	: adresse du boutli
POSITIONBOUTLI	: position du boutli
MEMPOINTENTITE	: variable de memorisation de la largeur de l'entite courante
POSITION	: variable de memorisation d'adresse d'un bout de ligne
TAMPON	: ensemble de caracteres constituant un mot
PENTITE	: pointeur vers une entite
PLIGNE	: pointeur vers une ligne
PBOUTLIGNE	: pointeur vers un bout de ligne
LETTRE	: un caractere du bout de ligne
SORTE	: type d'element typographique

2.2.2.1. PROCEDURE INIT

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procedure initialise toutes les variables de formatage

- Pré-condition

  RIEN

- Post-condition :

  (
  ensemble des variables mises à jour
)

- Procédure applelante :

  TRAITEMENTCLASSIQUE

- Procédure appelée :

  rien

- Algorithme logique :

  debut
  initialisation des variables à la valeur 0
  fin

- Spécification des variables :

  rien

2.2.1.2. PROCEDURE TRTMOT

- Entrées :

```
POINTSUCCESS : pointeur vers un mot
FO           : type de fonte
PO           : type de police
```

- Sorties :

```
NBCAR            : nombre de caracteres du mot
POSITION         : position courante sur le string mot
HAUTEURMOT       : hauteur du mot
TAMPON           : represente le string mot
LARGEURMOT       : largeur du mot
COMPTEURBLANCMOT : nombre de ' ' du mot
```

- Effet :

Cette procedure calcule la hauteur, la largeur d'un mot en fonction d'une police et d'une fonte determinée ainsi que le nombre de caractere ' ' qu'il contient

- Pre-conditions :

```
(
PO ^FO
)
```

- Post-conditions :

```
(
(0 <= NBCAR <= POSITION <= 24) ^
(largeur du mot deteminee de telle maniere qu'elle
 represente la somme des largeurs de tous les caracteres, y
 compris le(s) ' ') ^
(hauteur du mot determinee de telle maniere que ce soit
 la hauteur du plus grand caractere le composant exception faite
 de <,>,(,),[,],[,])
)
```

- Procedure appelante :

TRAITEMENTCLASSIQUE

- Procedure appelee :

rien

- Algorithme logique :

```
debut
  initialisation des variables ;
  tant qu'il reste des caracteres
    debut
      initialisation du placement du mot ;
      si le caractere = ' '
        debut
          incrementation du compteur de ' ' ;
          incrementation de la largeur du mot
        fin
      sinon
        debut
          examen de la hauteur et de largeur du caractere ;
          mise a jour de la largeur du mot
        fin
    fin
  mise a jour compteur du nombre de caractere du mot
fin
```

- Specification des variables :

```
I        : indice de boucle
HAUT     : hauteur du caractere
LARG     : largeur du caractere
TPLETTRE : caractere
```

2.2.1.3. PROCEDURE MISEAJOUR

- Entrees :

  HAUTBOUT : hauteur du bout de ligne
  HAUTLI : hauteur de la ligne
  LARGLI : largeur de la ligne
  LARGBOUT : largeur du bout de ligne
  HAUTMOT : hauteur du mot
  LARMOT : largeur du mot

- Sorties :

  idem mais mis à jour

- Effet :

  Cette procedure a pour but de mettre a jour la hauteur
  des lignes et des bouts de ligne en fonction de la hauteur du mot.
  Elle incremente egalement la largeur de la ligne et du bout de
  ligne en fonction du mot traite

- Pré-conditions :

  (
  HAUTBOUT ∧ HAUTLI ∧ LARGLI ∧ LARGBOUT ∧ HAUTMOT ∧ LARMOT
)

- Post-conditions

  (
  HAUTBOUT ∧ HAUTLI ∧ LARGLI ∧ LARGBOUT ∧ HAUTMOT ∧ LARMOT
)

- Procedure appelante :

  TRAITEMENTCLASSIQUE

- Procedure appelee :

  rien

- Algorithme logique :

```
debut
  mise a jour
fin
```

- Spécification des variables :

  rien

2.2.1.4. PROCEDURE TRAITEMENTCHANGEMENT

- Entrées :

NBCAR	: nombre de caracteres d'un mot
HAUTEURMOT	: hauteur d'un mot
LARGEURMOT	: largeur d'un mot
COMPTEURBLANC	: nombre de ' ' au sein de l'entité
COMPTEURBLANCBOUTLI	: nombre de ' ' dans un bout de ligne
COMPTEURBLANCLI	: nombre de ' ' dans une ligne
HAUTEURLI	: hauteur d'une ligne
HAUTEURBOUTLI	: hauteur d'un bout de ligne
LARGEURLI	: largeur d'une ligne
LARGEURBOUTLI	: largeur d'un bout de ligne
MEMLARGEURLI	: variable de mémorisation de la largeur d'une ligne
MEMLARGEURBOUTLI	: variable de mémorisation de la hauteur d'un bout de ligne
LARGEURBLANC	: largeur optimale du caractère ' '
MEMHAUTEURBOUTLI	: variable de mémorisation de la hauteur d'un bout de ligne
NBLI	: nombre de lignes
HAUTEUREN	: hauteur de l'entité
RESTE	: variable de mémorisation de la largeur d'une entité
NBBOUTLI	: nombre de bout de lignes
ADRESSEBOUTLI	: adresse du bout de ligne
POSITIONBOUTLI	: position du bout de ligne courant
PLIGNE	: pointeur vers une ligne
PBOUTLIGNE	: pointeur vers un bout de ligne
POSITION	: variable de mémorisation des adresses d'un bout de ligne
LARGBLANC	: largeur du ' ' fixée à priori
POINTMOT	: pointeur vers un mot
NBPOINTENTITE	: largeur d'une entité
INTERLIGNETYPO	: valeur de interligne
CENTRAGETYPO	: booléen indiquant si centrage ou non
FONTETYPO	: type de fonte
SOULIGNEMENTTYPO	: type de soulignement
POLICETYPO	: type de police
MARGECAUCHETYPO	: marge gauche
MARGEDROITE	: marge droite
TAMPON	: buffer contenant le mot

- Sorties :

idem, ces variables étant réinitialisées

- Effet :

Cette procédure recevant les paramètres du formatage, gère le placement des lignes et des bouts de ligne au sein de l'entité courante et crée un record de l'arbre de formatage correspondant à la ligne ou au bout de ligne. Elle gère le saut de ligne, la justification, et traite tout ce qui est changement, par rapport aux indications définies dans le format standard, au sein du bout de ligne courant. Elle retire également tous les blancs en fin de ligne et gère la dernière ligne non complète. La procédure renvoie tous les paramètres de formatage réinitialisés pour le traitement suivant

- Pré-conditions :

(
(tous les paramètres de formatage sont déterminés)
∧(POINTMOT <> NIL)
)

- Post-conditions :

(
(((RESTE < 0) ∧ (SAUT = TRUE)) ∧
((un bout de ligne et une ligne crées) ∧
(tous les éléments du record crée sont initialisés) ∧
(les ' ' en fin de ligne sont supprimés)))
∨((RESTE > 0) ∧ ((FONTETYPO <>) ∧ (POLICETYPO <>) ∧
(SOULIGNEMENTTYPO <>))
(un bout de ligne est crée et gèré)) ∧
((POINTMOT <> NIL) ∧ (bout de ligne gèré) ∧ (fin de ligne gerée))
)

- Procédure appelante :

TRAITEMENTCLASSIQUE

- Procédure appelées :

JUSTIFIE
GERERECORD
TRTFINLIGNE
TRTCOMPLETAGE
TRTCENTRAGE

- Algorithme logique :

```
début
  si la largeur standard est dépassée ou saut de ligne
    début
      remplissage du bout de ligne par les caractères le
          constituant ;
      élimination des caractères ' ' en fin de ligne ;
      si le mot ne saute pas, justification de la ligne ;
      création et gestion d'un bout de ligne ;
      réajustement de la coordonnée X et de la valeur de la
          largeur des ' ' du bout de ligne ;
      création et gestion d'une ligne ;
      gestion de la chaine des pointeurs lignes/bouts de ligne ;
      réinitialisation de toutes les variables de formatage
    fin
  sinon
    début
      si l'élement courant diffère du format standard
        début
          gestion et création d'un bout de ligne ;
          gestion de la chaine des bouts de ligne ;
          réinitialisation partielle des variables de formatage
        fin
      sélection du mot suivant ;
      si il n'y a pas de suivant
        début
          remplissage du bout de ligne par les caractères le
              constituant ;
          gestion du bout de ligne ;
          gestion de la fin de la ligne
        fin
    fin
fin
```

- Spécification des variables :

```
I             : indice de boucle
J             : indice de boucle
POS           : position dans le string mot
SORTE         : type de l'élement typographique
LETTRE        : caractère
TPHAUT        : hauteur d'un caractère
TPLAR         : largeur d'un caractère
MEMLIGNE      : pointeur vers une ligne
MEMBOUTLIGNE  : pointeur vers un bout de ligne
```

2.2.1.5. PROCEDURE JUSTIFICATION

- Entrées :

  LARGBLANC : largeur du ' ' fixée à priori
  PLIGNE : pointeur vers une ligne
  COMPTEURBLANCLI : nombre de ' ' dans une ligne

- Sortie :

  LARGEURBLANC : largeur optimale du ' '

- Effet :

  Cette procédure à partir du pointeur vers une ligne, de la largeur du ' ', et du nombre de ' ' de la ligne, détermine la largeur optimale du ' ' afin d'avoir une justification de la ligne à droite

- Pré-conditions :

  (
  (LARGBLANC > 0) ∧ COMPTEURBLANCLI ∧ (PLIGNE <> nil)
)

- Post-conditions :

  (
  (INTER <> nil) ∧
  (((LARGEURBLANC est fonction de la difference entre la largeur standard et effective d'une ligne) ∧ (COMPTEURBLANCLI > 0)))
  ∨((LARGEURBLANC = largeur fixée à priori)∧(COMPTEURBLNCLI = 0)))
  ∧(((nombre de caractères <> 0) ∧ (LARGEURBLANC = largeur moyenne d'un caractère))
  ∨((nombre de caractère = 0) ∧ (LARGEURBLANC = largeur fixée à priori)))
)

- Procédure appelante :

  TRAITEMENTCHANGEMENT

- Procédure appelée :

  rien

- Algorithme logique :

```
debut
  sélection du premier bout de ligne ;
  tant qu'il y a des bouts de ligne
    debut
      incrementation du nombre de caractères de la ligne ;
      incrementation de largeur de la ligne
    fin
  calcul de largeur de la ligne sans ' ' ;
  détermination de l'écart entre la largeur standard et la
        largeur effective ;
  calcul de la largeur du ' ' optimale
fin
```

- Spécification des variables :

CARA	: compteur du nombre de caractères d'une ligne
LIGNEUTILISEESANSBLANC	: largeur de ligne sans ' '
RESTE	: difference entre la largeur standard et la largeur effective d'une ligne
LARGCARA	: largeur moyenne d'un caractère
INTER	: pointeur vers un bout de ligne

2.2.1.6 PROCEDURE TRTFINLIGNE

- Entrées :

  PBOUTLIGNE : pointeur vers un bout de ligne
  PLIGNE : pointeur vers une ligne
  COMPTEURBLANCBOUTLI : nombre de ' ' d'un bout de ligne
  COMPTEURBLANCLI : nombre de ' ' d'un ligne
  LARGBLANC : largeur du ' ' donnée a priori

- Sorties :

  LARGEURBLANC : largeur optimal du ' '
  LARGEURLI : largeur d'une ligne

- Effet :

  Cette procédure traite la dernière ligne non complete
  et met à jour la largeur optimale du caractère ' ' et la largeur
  des lignes et des bouts de ligne

- Pré-conditions :

  (
  (PLIGNE <> NIL) ∧ (PBOUTLIGNE <> NIL) ∧ (LARGBLANC > 0)
)

- Post-conditions :

  (
  (LARGEURLI) ∧
  (pour tous les bouts de ligne de la ligne, leur largeur est ajustée) ∧
  (LARGEURBLANC)
)

- Procédure appelante

  TRAITEMENTCLASSIQUE

- Procédure appelée :

  rien

- Algorithme logique :

```
début
  sélection du premier bout de ligne ;
  tant qu'il reste des bouts de ligne
    début
      détermination de la largeur avec le(s) ' ' du bout de ligne ;
      détermination du nombre d'élément avec le(s) ' ' ;
      sélection du bout de ligne suivant
    fin
  initialisation ;
  détermination du nombre de caractère <> ' ' ;
  détermination de largeur moyenne d'un caractère ;
  détermination de la largeur de ligne ;
  si la largeur > largeur standard
    début
      calcul de l'écart entre largeur effective et voulue ;
      détermination de la valeur du ' ' optimale ;
      détermination de la largeur de la ligne
    fin
  sélection du premier bout de ligne ;
  tant qu'il reste des bouts de ligne
    début
      mise à jour de largeur du blanc ;
      mise à jour de la largeur
    fin
fin
```

- Spécification des variables :

  LARGEURAVECBLANC : largeur d'une ligne ou d'un bout de ligne avec ' '
  LARGEURSANSBLANC : largeur d'une ligne ou d'un bout de ligne sans ' '
  NOMBREELEMENTAVECBLANC : nombre de caractères d'une ligne ou d'un bout de ligne avec ' '
  NOMBREELEMENTSANSBLANC : nombre de caractères d'une ligne ou d'un bout de ligne sans ' '
  CARALARGMOYEN : largeur moyenne d'un caractère
  COMPTEURBLANCBOUTLI : nombre de ' ' du bout de ligne
  PBOUTLI : pointeur vers un bout de ligne

2.2.1.7 PROCEDURE TRTCOMPLETAGE

- Entrées :

PENTITE : pointeur vers l'entité
HAUTEUREN : hauteur de l'entité
INTERLIGNETYPO : espace entre les lignes

- Sortie :

rien

- Effet :

Cette procédure traite la fin d'un élément de structure en indiquant la hauteur et la position de chaque ligne et de chaque bout de ligne au sein de l'entité

- Pré-conditions :

(
(PENTITE <> NIL) ∧ (HAUTEUREN >0) ∧ (INTERLIGNETYPO >0)
)

- Post-conditions :

(
(PENTITE <> NIL) ∧
(les coordonnées X, Y des lignes sont mises à jour au sein de l'entité) ∧
(les coordonnées X, Y des bouts de lignes sont mises à jour au sein des lignes)
)

- Procédure appelante :

TRAITEMENTCLASSIQUE

- Procédure appelée :

rien

- Algorithme logique :

```
début
  sélection de la première ligne ;
  tant qu'il reste des lignes
    début
      initialisation de hauteur de l'entité ;
      calcul de la différence entre la largeur effective et la
          largeur standard ;
      détermination de la coordonnée Y de la ligne ;
      sélection du premier bout de ligne ;
      tant qu'il reste des bouts de lignes
        début
          calcul du nombre de . . ;
          si la largeur de(s) . . > différence entre la largeur effective
               et la largeur standard, mémorisation de cette
               différence et réinitialisation ;
          sinon
            début
              mémorisation du nombre de . . ;
              réinitialisation de la différence
            fin
          détermination des coordonnées X, Y des lignes et
                des bouts de lignes ;
          sélection de la ligne suivante
        fin
      mise à jour de la hauteur de l'entité ;
      sélection de la ligne suivante
    fin
fin
```

- Spécification des variables :

POS : variable de mémorisation de la hauteur d'une entité
VALX : coordonnée X d'une ligne et d'un bout de ligne
VARIABLE : variable contenant la différence entre la largeur effective et la largeur standard d'une ligne
NBBOUT : nombre de . . d'un bout de ligne
ACCROIT : facteur d'accroissement de la coordonnée X des bouts de ligne
PBOUTLIGNE : pointeur vers un bout de ligne
PLIGNE : pointeur vers une ligne

2.3. PROCEDURE MISEENPAGE

- Entrees :

MEMENTITE : pointeur de la première entité

PDOCUMENT : pointeur vers le document

- Sortie :

ADRPREMIEREPAGE : adresse de la première page créée

- Effet :

Cette procédure recevant le pointeur de la première entité et le pointeur vers le document, place ces entités de manière optimale sur la ou les page(s). Elle numérote éventuellement les pages et permet de n'imprimer qu'un nombre réduit de page(s) (nombre déterminé par l'utilisateur).

- Pré-conditions :

(
MEMENTITE $\wedge$ PDOCUMENT
)

- Post-condition :

(
ADRPREMIEREPAGE
)

- Procédure appelante :

ARBREFORMATAGE

- Procédures appelées :

SEULESURPAGE
UNECOLO
DEUXCOLO
BREAKPAGE
NUMEROTATION

- Algorithme logique :

```
debut
  si COLONNAGE = true
    debut
      si SEULESURPAGE = true
        debut
          mise en page des entités sur deux colonnes avec
          les éléments d'identifications seuls sur la
          première page
        fin
      sinon mise en page des entités sur deux colonnes
    fin
  sinon
    debut
      si SEULESURPAGE = true
        debut
          mise en page des entités sur une colonne avec
          les éléments d'identifications seuls sur
          la première page
        fin
      sinon mise en page des entités sur une colonne
    fin
  si NUMEROTATION = true les pages seront numérotées
  séquentiellement, la page contenant les éléments
  d'identifications ne sera pas numérotée
fin
```

- Spécification des variables :

ADRDERNIEREPAGECREE	: pointeur vers la dernière page du document
IEREPAGE	: indicateur booléen de première page
SEULESURPAGE	: booléen indiquant que les éléments d'identifications sont seuls sur une page

2.3.1. PROCEDURE UNECOLO

- Entrees :

ADRDERNIEREPAGECREEE	: pointeur vers la dernière page
ADRPREMIEREPAGE	: pointeur vers la première page
MEMENTITE	: pointeur vers la première entité
PDOCUMENT	: pointeur vers le document
SEULESURP	: booléen indiquant si les éléments d'identification sont centrés sur la première page
IEREPAGE	: booléen indiquant si c'est la première page

- Sorties :

ADRPERMIEREPAGE	: idem
ADRDERNIEREPAGECREE	: idem
MEMENTITE	: idem

- Effet :

Cette procédure détermine les coordonnées Y des entités pour tout le document en appliquant le principe de la "COLLE" décrit dans l'analyse fonctionnelle et renvoie le pointeur de la première et dernière page créée ainsi que le pointeur de la première entité. La mise en page est effectuée en simple colonne

- Pré-conditions :

```
(
(toutes les entités spécifiques au document sont chaînées) ∧
(((ADRPREMIEREPAGE = NIL) ∧ (SEULESURPAGE = FALSE) ∨ (IEREPAGE = TRUE)
 (ADRDERNIEREPAGECREE = NIL))
∨ ((ADRPREMIEREPAGE = ADRDERNIEREPAGECREE) ∧
 (SEULESURPAGE = TRUE) ∧ (IEREPAGE = FALSE)))
∧ ((MEMENTITE <> NIL)   (PDOCUMENT <> NIL))
)
```

- Post-conditions :

```
(
((ADRPREMIEREPAGE <> NIL) ∧ (ADRDERNIEREPAGECREE <> NIL)
 (IEREPAGE = FALSE) ∧ (SEULESURAGE = TRUE) ∧ (SEULESURPAGE = FALSE))∧
(toutes les entités spécifiques au document sont chaînées
 de telle manière que les notes succèdent les autres types
 d'entités)∧
(tous les paramètres de la mise en page sont déterminés) ∧
(aucun chevauchement des entités et les coordonnées Y sont
 déterminées de telle manière que : soit y1, y2 les
 coordonnées respectivement de entité 1 et entité 2 et
 l'entité 1 précédant l'entité 2, la relation :
 y2 = y1 - longueur entité 1 - écart est toujours vérifiée ;
 écart étant le facteur d'élasticité)∧
(ECARTMINIMUM < ECARTOPTIMAL < ECARTMAXIMUM)
)
```

- Procédure appelante :

MISEENPAGE

- Procédures appelées :

UPOSITIONY1
UPOSITIONY2
UPOSITIONY3
UCREERPAGE
URESERVATIONNOTE
UPLACEMENTNOTE
UCASSURZ

- Algorithme logique :

```
début
  initialisation des paramètres de mise en page pour tout le document ;
  tant qu'il y a des entités
    début
      initialisation des paramètres de mise en page pour une page ;
      tant qu'il reste de la place et des entités
        si l'entité doit sauter de page et que ce n'est pas la première
          début
            initialisation des variables pour le saut ;
            gestion de la chaine des entités
          fin
        sinon
          début
            si le type d'entité est une note, gestion de la réservation ;
            sinon
              début
                décrémentation de la place restante ;
                gestion de la chaine des entités ;
                sélection de l'entité suivante
              fin
          fin
    fin
  si la page n'est pas complètement remplie
    début
      gestion du placement des notes ;
      gestion du placement des entités ;
      création d'une page ;
      si une entité a du sauter de page, réinitialisation des
        variables de saut
    fin
  sinon
    début
      si il ne reste plus de place sur la page
        début
          calcul d'une variable résultat afin de déterminer
             si l'entité chevauchante peut être placée
             sur la page courante ;
          si résultat >= 0
            début
              calcul d'un écart afin de placer toutes les entités;
              détermination des coordonnées Y des entités de
                  la page ;
```

```
          reajustement des coordonnees afin d'obtenir un
              remplissage optimal de la page ;
          placement des notes ;
          creation de la page
        fin
      sinon
        debut
          calcul d'une variable resultat afin de determiner
              si l'entite chevauchante doit etre
              placee en partie sur la page courante
              ou sur la page suivante ;
        si resultat <= 0
          debut
            calcul d'un ecart pour placer toutes les entites ;
            determination des coordonnees Y des entites de la page ;
            reajustement des coordonnees afin d'obtenir un
                rempissage optimal de la page ;
            placement des notes ;
            creation d'une page
          fin
        sinon
          debut
            placement des notes ;
            gestion de la cassure de l'entite chevauchante ;
            creation d'une page ;
            gestion de la chaine des entites
          fin
    fin
  fin
 fin
fin
```

- Specification des variables :

ECARTMINIMUM	: ecart minimum entre entites
ECARTMAXIMUM	: ecart maximum entre entites
MARGESUPERIEURE	: marge superieure d'une page du document
MARGEINFERIEURE	: marge inferieure du document
LONGUEURDISPONIBLE	: place restante sur la page
NBENTITEPAGE	: compteur du nombre d'entites sur la page
COMPTEUR	: compteur intermediaire du nombre d'entites sur la page
INTERMEDIAIRE	: facteur d'elasticite dans le cas du serrage
LONGUEURNOTE	: place reservee pour les notes
LONGPOIPA	: longueur d'une feuille du document
LARPOIPA	: largeur d'une feuille du document
ECARTMOYEN	: ecartmoyen entre entite
DISTANCE	: espace netre la dernière entite placee et le bas de la page
EACRTOPTIMAL	: facteur d'elasticite dans le cas de la cassure
RESTENOTE	: variable intermediaire indiquant la place reservee pour les notes
ADRPREMIERENOTE	: pointeur vers la premiere note
MEMOIREENTITE	: pointeur vers l'entite precedant une note
MPADRNOTE	: pointeur vers la dernière note
ENTITECOURANTE	: pointeur vers l'entite courante
MEMORY	: pointeur de memorisation de l'entite courante
PTENTITE	: pointeur vers l'entite creee
PTPAGE	: pointeur vers la page courante
MPPROCHAIN	: pointeur vers la dernière entite de la page dans le cas du reajustement
MEMSAUT	: pointeur vers l'entite precedant celle qui saute de page
TEST	: variable booleenne indiquant si il reste des entites a traiter
IERNOTE	: variable booleenne indiquant la premiere occurrence d'une note
SAUT	: variable booleenne indiquant si une entite doit sauter de page

2.3.1.1. PROCEDURE UPOSITIONY1

- Entrées :

ENTITECOURANTE	: pointeur vers l'entité courante
MEMENTITE	: pointeur vers la première entité de la page
LONGUEURDISPONIBLE	: place restante sur la page
LONGPOIPA	: longueur de la feuille du document
ECARTMOYEN	: écart moyen entre entité
MARGESUPERIEURE	: marge supérieure d'une feuille du document
LONGUEURNOTE	: place réservée aux notes bas de page

- Sortie :

rien

- Effet :

Cette procédure détermine les coordonnées Y des entités sélectionnées pour la page courante dans le cas où le facteur élasticité est nul et où la page n'est pas remplie. L'espace entre les entités de la page est simplement un écart moyen standard fixé à priori. Les coordonnées sont déterminées en fonction de la longueur d'une feuille, du placement des autres entités et de la place réservée au(x) note(s) bas de page si elle(s) existe(nt)

- Pré-conditions :

(
((MEMENTITE <> NIL) ∧ (ENTITECOURANTE <> NIL) ∧
(longueur des entités sélectionnées pour la page <=
LONGUEURDISPONIBLE <= (LONGPOIPA - (MARGESUPERIEURE - LONGUEURNOTE))) ∧
(LONGPOIPA ECARTMOYEN MARGESUPERIEURE) ∧
(MARGEINFERIEURE <= LONGUEURNOTE < LONGPOIPA) ∧
(entités à traiter sont chainées)
)

- Post-conditions :

(
(chainage des entités de la page courante) ∧
(chaque entité a sa coordonnée Y déterminée de telle manière que : soit y1 la coordonnée de l'entité 1, y2 la coordonnée de entité 2 et l'entité 1 précédant l'entité 2 la relation : y2 = y1 - longueur entité 1 - ecartmoyen est toujours vérifiée) ∧
(aucun chevauchement des entités) ∧
(LONGUEURDISPONIBLE >= LONGUEURNOTE)
)

- Procédure appelante :

UNECOLO

- Procédure appelée :

rien

- Algorithme logique :

```
début
  si la première entité est une note, sélection de
       l'entité suivante ;
  initialisation de l'espace réservé aux entités ;
  tant qu'il reste des entités
    début
      détermination de la coordonnée Y ;
      mise à jour de la place disponible ;
      sélection de l'entité suivante
    fin
fin
```

- Spécification des variables :

rien

2.3.1.2. PROCEDURE UPOSITIONY2

- Entrées :

ENTITECOURANTE	: pointeur vers l'entité courante
DISTANCE	: longueur entre la dernière entité placée et le bas de la feuille
MPCOMPTEUR	: nombre d'entités placées sur la page courante
MEMENTITE	: pointeur vers la première entité de la page
LONGUEURDISPONIBLE	: place restante sur la page
MARGESUPERIEURE	: marge supérieure d'une feuille du document
MPINTERMEDIAIRE	: facteur d'élasticité
NBENTITEPAGE	: nombre d'entités sélectionnées pour la page courante
LONGUEURNOTE	: place réservée aux notes bas de page

- Sorties :

ENTITECOURANTE	: pointeur vers l'entité courante
DISTANCE	: longueur entre la dernière entité placée et le bas de la feuille
MPCOMPTEUR	: nombre d'entités placées sur la page

- Effet :

Cette procédure détermine les coordonnées Y des entités sélectionnées pour la page dans le cas d'un chevauchement. Une entité différente d'une note ou d'une figure chevauche sur deux pages et en resserrant les entités de la page courante, l'entité chevauchante peut être insérée sans devoir être cassée et en placer une partie sur la page suivante. Après avoir placé toutes les entités sélectionnées sur la page courante, cette procédure réajuste celles-ci afin d'obtenir un remplissage optimal de la page.

- Pré-conditions :

(
((ENTITECOPANTE <> NIL) (MEMENTITE <> NIL)) ∧
(LONGUEURNOTE <= DISTANCE <= longueur des entités courantes) ∧
(0 <= MPCOMTEUR <= NBENTITEPAGE) ∧
(LONGUEURDISPONIBLE ∧ LONGPOIPA ∧ MARGESUPERIEURE) ∧
((MPINTERMEDIAIRE représente l'élasticité entre entités ;
il exprime le fait qu'en resserrant les entités sur la page,
l'entité chevauchante peut être placée) ∧
(MPINTERMEDIAIRE = ECARTMINIMUM augmenté du gain de place
dû au serrage) ∧
(MPINTERMEDIARE représente l'écart entre entité
afin d'avoir un remplissage optimal de la page)) ∧
(MARGEINFERIEURE <= LONGUEURNOTE <= LONGPOIPA)∧
(NBENTITEPAGE > 1)∧
(toutes les entités sont chainées)
)

- Post-conditions :

(
(chainage des entités traitées et spécifiques à la page
courante)∧
(aucun chevauchement entre entités) ∧
(LONGUEURDISPONIBLE >= LONGUEURNOTE)∧
((NBENTITEPAGE = 0) (MPCOMPTEUR = nombre d'entités sur la page))∧
((chaque entité sélectionnée a sa coordonnée Y
déterminée de telle manière que : soit y1 la coordonnée
de entité 1, y2 la coordonnée de entité 2 et l'entité 1
précédant l'entité 2, la relation y2 := y1 - longueur de
l'entité 1 - mpintermédiaire soit toujours vérifiée)∧
(DISTANCE = (MPDERNIERE^.Y - LONGUEURNOTE)) > 0)∧
((idem pour la proposition 1) ∧
(DISTANCE = (MPDERNIERE^.Y - LONGUEURNOTE)) = 0)
)

- Procédure appelante :

UNECOLO

- Procédure appelée :

rien

- Algorithe logique :

```
debut
  tant que le nombre d'entités > 0
    debut
      mémorisation de la dernière entité placée ;
      détermination des coordonnées Y de l'entité ;
      mise à jour de la pace restante, du nombre d'entités
            et de la distance ;
    fin
  mise à jour de la chaine des entités
fin
```

- Spécification des variables :

MPDERNIERE : pointeur vers la dernière entité de la page courante

2.3.1.3. PROCEDURE UPOSITIONY3

- Entrées :

ENTITECOURANTE	: pointeur vers l'entité courante
DISTANCE	: longueur entre la dernière entité placée et le bas de la feuille
MPCOMPTEUR	: nombre d'entités placées sur la page courante
MEMENTITE	: pointeur vers la première entité de la page
LONGUEURDISPONIBLE	: place restante sur la page
MARGESUPERIEURE	: marge supérieure d'une feuille du document
MPINTERMEDIAIRE	: facteur d'élasticité
NBENTITEPAGE	: nombre d'entités sélectionnées pour la page courante
LONGUEURNOTE	: place réservée aux notes bas de page

- Sorties :

ENTITECOURANTE	: pointeur vers l'entité courante
DISTANCE	: longueur entre la dernière entité placée et le bas de la feuille
MPCOMPTEUR	: nombre d'entités placées sur la page

- Effet :

Cette procédure détermine les coordonnées Y des entités sélectionnées pour la page dans le cas d'un chevauchement. L'entité chevauchante ne pouvant être placée sur la page courante même en resserrant les entités précédantes. Elle ne peut non plus être placée en partie étant donné un critère de taille. Cette procédure renvoie cette entité à la page suivante, place les entités précédantes sur la page courante et réorganise le placement afin d'obtenir un remplissage optimal de la page

- Pré-conditions :

(
((ENTITECORANTE <> NIL) ∧ (MEMENTITE <> NIL)) ∧
(LONGUEURNOTE <= DISTANCE <= longueur des entités courantes) ∧
(0 <= MPCOMTEUR <= (NBENTITEPAGE - 1)) ∧
(LONGUEURDISPONIBLE LONGPOIPA MARGESUPERIEURE) ∧
((MPINTERMEDIAIRE représente l'élasticité entre entités ;
il exprime le fait qu'en resserrant les entités sur la page,
l'entité chevauchante peut être placée) ∧
(MPINTERMEDIAIRE = ECARTMINIMUM augmenté du gain de place
dû au serrage) ∧
(représente l'écart entre entité afin d'avoir un remplissage optimal de la PAGE)) ∧
(MARGEINFERIEURE <= LONGUEURNOTE <= LONGPOIPA) ∧
(NBENTITEPAGE > 1) ∧
(toutes les entités sont chainées)
)

- Post-conditions :

(
(chainage des entités traitées et spécifique à la page courante) ∧
(aucun chevauchement entre entités) ∧
(LONGUEURDISPONIBLE >= LONGUEURNOTE) ∧
((NBENTITEPAGE = 1) ∧ (MPCOMPTEUR = nombre d'entité sur la page)) ∧
((chaque entité sélectionnée moins une, a sa coordonnée Y déterminée de telle manière que : soit y1 la coordonnée de entité 1, y2 la coordonnée de entité 2 et l'entité 1 précédant l'entité 2, la relation y2 := y1 - longueur de l'entité 1 - mpintermédiaire soit toujours vérifiée) ∧
(DISTANCE = (MPDERNIERE^.Y - LONGUEURNOTE)) > 0) ∧
((idem pour la proposition 1) ∧
(DISTANCE = (MPDERNIERE^.Y - LONGUEURNOTE)) = 0)
)

- Procédure appelante :

UNECOLO

- Procédure appelée :

rien

- Algorithe logique :

```
début
  tant que le nombre d'entités > 0
    début
      mémorisation de la dernière entité placée ;
      détermination des coordonnées Y de l'entité ;
      mise à jour de la place restante, du nombre d'entités
            et de la distance ;
    fin
  mise à jour de la chaine des entités
fin
```

- Spécification des variables :

MPDERNIERE : pointeur vers la dernière entité de la page courante

2.3.1.4. PROCEDURE UCREERPAGE

- Entrées :

ADRPREMIEREPAGE	: pointeur vers la première page
ADRDERNIEREPAGECREEE	: pointeur vers la dernière page
IEREPAGE	: booléen indiquant si une page a déjà été créée pour le document
PTPAGE	: pointeur vers la page créée
ADRPREMIERENOTE	: pointeur vers la première note:
MEMENTITE	: pointeur vers la première entité de la page

- Sorties :

ADRPREMIEREPAGE	: idem
ADRDERNIEREPAGECREEE	: idem
IEREPAGE	: idem
PTPAGE	: idem

- Effet :

Cette procédure crée une page et met à jour les pointeurs de la page et de la chaine des pages : elle crée la chaine des pages et relie la page courante à la précédante, établit le lien entre la page créée et ses entités, et relie la dernière entité de la page à la première note bas de page si elle existe

- Pré-conditions :

```
(
((IERENOTE ) ∧
 (ADRPREMIEREPAGE = NIL) ∧ (ADRDERNIEREPAGECREE = NIL)) ∧
(MEMENTITE <> NIL)∧
(ADRPREMIERENOTE )
)
```

- Post-conditions :

```
(
((IERNOTE = FALSE) ∧(ADRPREMIEREPAGE = NIL)
 (ADRDERNIEREPAGECREEE = PTPAGE)) ∧
(toutes les entités spécifiques à une page sont chainées) ∧
(toutes les notes bas de page spécifiques à une page sont ∧
 chainées et la première est reliée à la dernière entité)
)
```

- Procédures appelantes :

SEULESURPAGE
UNECOLO

- Procédure appelée :

RIEN

- Algorithme logique :

```
début
  création d'une page ;
  si c'est la première page du document
    création de la chaine des pages ;
  sinon
    mise à jour de la chaine des pages
  mise à jour des éléments de la page dont la filiation
         page - entité ;
  tant qu'il reste des entités pour cette page
    début
      mémorisation de l'entité ;
      sélection de l'entité suivante
    fin
  lien de la dernière entité et de la première note

fin
```

- Spécification des variables :

MPMEM	: pointeur intermédiaire vers une entité
MPMEMOIRE	: pointeur de mémorisation de la dernière entité de la page

2.3.1.5. PROCEDURE BREACKPAGE

- Entrées :

  ADRPREMIERPAGE : pointeur vers la premiere page
  PDOCUMENT : pointeur vers le document

- Sortie :

  rien

- Effet :
  Cette procedure a pour but de sélectionner un nombre de pages afin de permettre d'imprimer uniquement ces pages. La borne superieure et inferieure du nombre ont été définies par l'utilisateur

- Pré-conditions :

  (
  PREMIERE <= DERNIERE $\wedge$ ADRPREMIEREPAGE .
)

- Post-conditions :

  (
  (ADRPREMIEREPAGE = PREMIERE) $\wedge$ (ADRDERNIEREPAGE = DERNIERE)
)

- Procédure appelante :

  MISEPAGE

- Procédure appelee :

  rien

- Algorithme logique :

```
debut
  tant que NBPAGE <= PREMIERE
    lecture de la page suivante
  si PREMIERE = DERNIERE sélection d'une seule page
  sinon
    debut
      tant que NBPAGE <= DERNIERE
        debut
          lecture de la page suivante ;
          selection de la dernière page ;
          mise à jour des pointeurs de la chaine des pages
        fin
    fin
fin
```

- Spécification des variables :

  POINTEURPAGE : pointeur vers une page
  MEMPOINTEUPAGE : pointeur intermédiaire vers uné page
  NBPAGE : variable entière indiquant le nombre de pages du document

2.3.1.6. PROCEDURE URESERVATIONNOTE

- Entrées :

LONGUEURNOTE	: place réservée sur la page pour les notes
ECARTMINIMUM	: écart minimum entre entité
ADRPREMIERENOTE	: pointeur vers la première note
ENTITECOURANTE	: pointeur vers l'entité courante
MEMOIREENTITE	: pointeur vers l'entité précédant la ou les note(s)
MPADRNOTE	: pointeur intermédiaire vers une note
IERNOTE	: booléen indiquant la première occurrence d'une note
LONGUEURDISPONIBLE	: place restante disponible sur la page

- Sorties :

idem sauf ECARTMINIMUM
Toutes ces variables sont mises à jour et ENTITECOURANTE devient le pointeur suivant la dernière note

- Effet :

Cette procédure a pour but de réserver de la place pour la note sur la page courante dans le cas du simple colonnage.
La logique de cette procédure est la suivante : toutes les entités sont chainées : c'est ce que nous appelons chainage classique ; ces entités sont toutes les composantes de l'arbre de formatage. A chaque rencontre d'une entité note, la procédure la place en considérant pour celle-ci un espace entre les autres entités ; espace différent suivant que c'est la première note ou non. Elle crée ensuite une chaine qui ne reprend que les notes et détache celle - ci de la chaine classique.

- Pré-conditions :

(
(((ADRPREMIERENOTE = NIL) ∧ (IERNOTE = TRUE) ∧ (MPADRNOTE = NIL)) ∨
((ADPREMIERENOTE = NIL) ∧ (IERNOTE = FALSE) ∧ (MPADRNOTE <> NIL)))
(MEMOIREENTITE <> note) ∧
(0 <= LONGUEURNOTE <= LONGUEURFEUILLE) ∧
(largeur de l'entité note + ECRTMINIMUM <= LONGUEURDISPONIBLE
<= LONGUEURFEUILLE) ∧
(chainage de toutes les entités classiques et des notes
éventuelles)
)

- Post-conditions :

(
(mise à jour de la chaine classique des entités,
MEMOIREENTITE contenant l'adresse de l'entité suivant la note) ∧
(mise à jour de la chaine des notes) ∧
(((IERNOTE = FALSE) ∧ (MPADRNOTE = ENTITECOURANTE)) ∨
((IERNOTE = FALSE) ∧ (MPADRNOTE = ENTITECOURANTE)
(ADRPREMIERENOTE = ENTITECOURANTE))) ∧
(((0 <= LONGUEURDISPONIBLE <= LONGUEURFEUILLE - (LONGUEURNOTE -
3 * ECARTMINIMUM)) ∧ ((IERNOTE = TRUE) ∨ (IERENOTE = FALSE))))
(((LONGUEURNOTE + ECARTMINIMUM) <= LONGUEURNOTE <= LONGUEURFEUILLE) ∧

(aucun chevauchement ni horizontal ni vertical de la note avec les
autres entités)
)

- Procédure appelante :

DEUXCOLO

- Procédure appelée :

rien

- Algorithme logique :

```
debut
  si c'est la premiere note
    debut
      tant qu'il y a des notes
        debut
          incrementation de la place reservee aux notes ;
          decrementation de la place restante disponible pour
              les autres entites ;
          creation de la chaine des notes ;
          mise a jour de la chaine classique des entites ;
          selection de la note suivante
        fin
    fin
  sinon
    debut
      tant qu'il y a des notes
        debut
          mise a jour de la chaine des notes ;
          incrementation de la place reservee pour les notes ;
          decrementation de la place restante disponible pour
                les autres entites ;
          mise a jour de la chaine classique des entites ;
          selection de la note suivante
        fin
    fin
fin
```

- Spécification des variables :

Rien

2.3.1.7. PROCEDURE UPLACEMENTNOTE

- Entrées :

RESTENOTE : place allouée sur la page pour les notes
ECARTMINIMUM : écart minimum entre entités
ADRPREMIERENOTE : pointeur vers la première note
PDOCUMENT : pointeur vers le document

- Sorties :

ADRPREMIERENOTE
PDOCUMENT

- Effet :

Cette procédure, connaissant l'adresse de la première note et la place réservée pour celle-ci, arrange les notes en fin de page dans le cas du simple colonnage et crée une séparation entre la ou les note (s) et les autres entités

- Pré-conditions :

(
(LONGUEURFEUILLE > RESTENOTE > 0),
∧(ADRPREMIERENOTE <> NIL)
∧(toutes les notes sont chainées)
)

- Post-conditions :

(
(RESTENOTE = 0)
∧(chaque note a sa coordonnée Y déterminée)
∧(toutes les notes sont chainées)
∧(écart entre les entités de type note = ECARTMINIMUM)
∧(la ou les note(s) ne chevauche(nt) pas les autres entités)
∧(toutes les notes selectionnées sont placées)
∧(elles sont placées par ordre décroissant de leur référence)
)

- Procédure appelante :

UNECOLO

- Procédures appelées :

EXAMEN
CREERECORD
GERERRECORD

- Algorithme logique :

```
début
  création des records pour construire la séparation entre
      les notes et les autres entités ;
  gestion de l'entité créée ;
  selection de la première note ;
  placement de la coordonnée X de la note ;
  tant qu'il y a des notes
    début
      placement de la coordonnée X de la note ;
      décrémentation de la place allouée à la note ;
    fin

  selection de la note suivante
fin
```

- Spécification des variables :

PENTITE	: pointeur vers l'entité de séparation créée
PLIGNE	: pointeur vers la ligne de séparation créée
PBOUTLI	: pointeur vers le bout de ligne de séparation crée
INDICENOTE	: pointeur vers une note
INTERMEDIAIRE	: pointeur de mémorisation d'adresse d'une note
POSITIONX	: variable entière indiquant la coordonnée X de l'entité séparation par rapport à la page
POSITIONY	: idem mais coordonnée Y
LARGMOT	: variable entière contenant la largeur de la ligne de séparation
I	: variable entière d'indice de boucle
LARGEUR	: paramètre pour la procédure EXAMEN
HAUTEUR	: idem
CARA	: caractère "-" constituant la séparation
SORTE	: variable déterminant le type d'élement typographique

2.3.1.8. PROCEDURE UCASSURE

- Entrées :

PTENTITE	: pointeur vers l'entité créée
MEMENTITE	: pointeur vers la première entité de la page
ENTITECOURANTE	: pointeur vers l'entité courante
LONGUEURNOTE	: place réservée aux notes
LONGUEURDISPONIBLE	: place restante disponible sur la page
ECARTMOYEN	: espace moyen entre entités
MARGESUPERIEURE	: marge supérieure d'une feuille du document
NBENTITEPAGE	: nombre d'entités placées sur la page
LONGPOIPA	: longueur d'une feuille du document

- Sorties :

PTENTITE	: idem
ENTITECOURANTE	: idem
LONGUEURDISPONIBLE	: idem

- Effet :

Cette procédure place les entités sur la page courante jusqu'à concurrence de place disponible. Lorsqu'il ne reste plus assez de place pour ajouter une entité, la procédure gère la cassure de celle-ci en s'assurant que ce n'est pas une entité de type figure. Pour casser l'entité, la procédure place les lignes de l'entité jusqu'à concurrence de place disponible et crée une nouvelle entité de même type que celle cassée en lui chainant les lignes n'ayant pu être placées sur la page courante.
La procédure gère ensuite la page courante en réorganisant les coordonnées des entités déjà placées afin d'avoir un remplissage optimal de la page ; elle redéfini les coordonnées Y des lignes et des boutlignes au sein de l'entité chevauchante et de l'entité créée. Si l'entité chevauchante est de type figure, elle est reportée à la page suivante

- Pré-conditions :

(
((MEMENTITE <> NIL) ∧(PTENTITE <> NIL) ∧(ENTITECOURANTE <> NIL)) ∧
(NBENTITEPAGE > 1)∧
(toutes les entités sont chainées)
)

- Post-conditions :

(
((ENTITECOURANTE <> NIL) ∧ (PLACE > 0)∧(entite <> figure)
∧(les entités sont placées de telle manière que : soit
y1 la coordonnée de l'entité 1, y2 la coordonnée de
l'entité 2 et l'entité 1 précédant l'entité 2, la
relation : y2 = y1 - longueur entité 1 - écartmoyen est
toujours vérifiée)
∧(PTENTITE <> NIL)
∧(les lignes et les bouts de ligne sont réarangés au sein de l'entité
courante de telle manière que soit y1 la coordonnée de
la ligne 1, y2 la coordonnée de la ligne 2, et la ligne 1
précédant la ligne 2, la relation : y2 = y1 - largeur ligne 1
- interligne, soit toujours vérifiée ; même critère
pour les boutlignes)
∧(même disposition au sein de l'entité créée)
∧(largeur ligne < largeur ENTITECOURANTE < LONGUEURDISPONIBLE)
∧(les entités de la page courante sont chainées))
∨((ENTITECOURANTE <> NIL) ∧ (PLACE > 0) ∧ (entite = figure)
∧(réarangement des autres entités de la page de telle manière
que : soit y1 la coordonnée de l'entité 1, y2 la coordonnée
de l'entité 2, et l'entité 1 précédant l'entité 2, la
relation : y2 = y1 - largeur entité 2 - écartoptimal, est
toujours vérifiée ; l'écartoptimal représentant le
facteur élasticité afin d'obtenir un remplissage optimal
de la page)
∧(PTENTITE = ENTITECOURANTE)
∧(les entités spécifiques à la page sont chainées))
)

- Procédure appelante :

UNECOLO

- Procédure appelée :

rien

- Algorithme logique :

```
début
  tant qu'il reste des entites
    début
      si il reste de la place
        debut
          determination des coordonnées y de l'entité ;
          mise à jour de la place restante sur la page ;
          selection de l'entité suivante
        fin
      sinon
        si l'entité <> figure
          début
            tant qu'il reste de la place
              début
                determination des coordonnées Y des lignes ;
                incrémentation du compteur des lignes ;
                mémorisation de la ligne ;
                sélection de la ligne suivante
              fin
            tant que (compteur de ligne - 1) > 0
              début
                mise à jour des coordonnées Y des lignes ;
                selection de la ligne suivante
              fin
            mise à jour de la chaine des pointeurs des lignes
                et des entités ;
            creation d'une nouvelle entité ;
            mise à jour de la longueur de l'entité créée ;
            mise à jour des coordonnées Y des lignes et des
                bouts de ligne au sein de l'entité créée ;
            mise à jour des différents éléments de la
                nouvelle entité
          fin
        sinon
          début
            selection de l'entité figure pour la page suivante ;
            tant qu'il reste des entités
              début
                determination des nouvelles coordonnées Y des
                        entités ;
                mise à jour de la distance entre l'entité placée
                        et le bas de page ;
                selection de l'entité suivante
              fin
          fin
  fin
fin
```

- Spécification des variables :

MEMOIRELIGNE	: pointeur de mémorisation de la dernière ligne de l'entité cassée
MEMLIGNE	: pointeur de mémorisation de la première ligne de l'entité créée
MEMOIREFIGURE	: pointeur de la dernière entité placée sur la page
LIGNE	: pointeur intermédiaire vers une ligne
BOUTLIGNE	: pointeur intermédiaire vers un bout de ligne
TEST	: place restante disponible sur la page pour placer une partie de l'entité chevauchante
NBENT	: indique le nombre d'entités de la page
ECARTOPTIMAL	facteur d'élasticité des entités pour la page courante
DISTANCE	: variable entière contenant l'espace entre la coordonnée Y de la dernière entité et le bas de la page
DECR	: variable entière de mise à jour des coordonnées Y des lignes et des boutlignes au sein de l'entité cassée et créée
CLIGNE	: compteur du nombre de lignes de l'entité chevauchante

2.3.2. PROCEDURE DEUXCOLO

- Entrées :

ADRDERNIEREPAGECREE	: pointeur vers la dernière page
ADRPREMIEREPAGE	: pointeur vers la première page
MEMENTITE	: pointeur vers la premiere entité
PDOCUMENT	: pointeur vers le document
SEULESURP	: booléen indiquant si les éléments d'identification sont centrés sur la première page
IEREPAGE	: booléen indiquant si c'est la première page

- Sorties :

ADRPERMIEREPAGE	: idem
ADRDERNIEREPAGECREE	: idem
MEMENTITE	: idem

- Effet :

Cette procédure détermine les coordonnées X, Y des entités pour tout le document en appliquant le principe de la "glu" décrit dans l'analyse fonctionnelle et renvoie le pointeur de la première et dernière page créée ainsi que le pointeur de la première entité. La mise en page est effectuée en double colonne

- Pre-conditions:

(
(toutes les entités spécifiques au document sont chainées) ∧
(((ADRPREMIEREPAGE = NIL) ∧ (SEULESURPAGE = FALSE) ∧ (IEREPAGE = TRUE)
(ADRDERNIEREPAGECREE = NIL)) ∨
((ADRPREMIEREPAGE = ADRDERNIEREPAGECREE) ∧
(SEULESURPAGE = TRUE) ∧ (IEREPAGE = FALSE))) ∧
((MEMENTITE <> NIL) ∧ (PDOCUMENT <> NIL))
)

- Post-conditions :

(
((ADRPREMIEREPAGE <> NIL) ∧ (ADRDERNIEREPAGECREE <> NIL) ∧
(IEREPAGE = FALSE) ∧ (SEULESURPAGE = TRUE) ∨ (SEULESURPAGE = FALSE)) ∧
(toutes les entités spécifiques au document sont chainées
de telle manière que les notes succèdent les autres types
d'entités ; les figures succèdent les autres entités mais
précèdent les notes) ∧
(tous les paramètres de la mise en page sont déterminés) ∧
(aucun chevauchement des entités et les coordonnées Y sont
déterminées de telle manière que : soit y1, y2 les
coordonnées respectivement de entité 1 et entité 2 et
l'entité 1 précédant l'entité 2, la relation :
y2 = y1 - longueur entité 1 - écart est toujours vérifiée ;
écart étant le facteur d'élasticité) ∧
(les coordonnées X sont déterminées de telle manière
que : soit x1, x2 les coordonnées respectivement de entité 1 et
entité 2, l'entité 1 étant placée sur la première
colonne et l'entité 2 étant placée sur la deuxième
colonne, la relation : x1 = 0 et x2 = 945 est toujours vérifiée) ∧
(ECARTMINIMUM < ECARTOPTIMAL < ECARTMAXIMUM)
)

- Procédure appelante :

MISEENPAGE

- Procédures appelées :

RESERVATIONNOTE
RESERVATIONFIGURE
POSITIONY1
PLACEMENTFIGURE
PLACEMENTNOTE
SAUTPAGE
COLONNAGE
CASSURE

- Algorithme logique :

```
début
  initialisation des paramètres de mise en page pour tout le document ;
  tant qu'il y a des entités
    début
      initialisation des paramètres de mise en page pour une page ;
      tant qu'il reste de la place et des entités
        début
          si l'entité doit sauter de page et que ce n'est pas la première
            initialisation des variables pour le saut ;
            gestion de la chaine des entités
          fin
        sinon
          début
            si le type d'entité est une note, gestion de la réservation ;
            sinon
              si le type d'entité est une figure, réservation de la place
              sinon
                si le type d'entité est un élément d'identification
                  début
                    réservation de la place ;
                    gestion de la chaine des entités
                  fin
                sinon
                  début
                    décrémentation de la place restante ;
                    gestion de la chaine des entités ;
                    sélection de l'entité suivante
                  fin
          fin
    fin
  si la page n'est pas complètement remplie
    début
      détermination des coordonnées Y des entités sélectionnées ;
      placement des figures éventuelles ;
      placement des notes éventuelles ;
      création d'une page ;
      si l'entité doit sauter de page
        début
          réinitialisation des variables de saut ;
          gestion de la chaine des entités
```

```
      fin
  fin
    sinon
      debut
        si une seule entite a ete selectionnee
          debut
            gestion du passage partiel de l'entite de la premiere
                a la deuxieme colonne ;
            gestion de la chaine des entites ;
            gestion du passage partiel de l'entite de la page courante
                a la page suivante ;
            placement des figures eventuelles ;
            placement des notes eventuelles ;
            creation d'une page
          fin
sinon
  debut
    placement de toutes les entites selectionnees sauf la derniere ;
    si la derniere entite placee est sur la deuxieme colonne
      debut
        initialisation de la place restante ;
        si la place n'est pas suffisante pour en placer une partie
          debut
            gestion du passage partiel de la derniere entite
                de la page courante a la suivante ;
            placement des figures eventuelles ;
            placemant des notes eventuelles ;
            gestion de la chaine des entites ;
            creation d'une page
          fin
        sinon
          debut
            placement des figures eventuelles ;
            placement des notes eventuelles ;
            creation d'une page ;
            gestion de la chaine des entites
          fin
      fin
    sinon
      debut
        initialisation de la place restante ;
        gestion du passage partiel de la derniere entite
            placee de la premiere a la deuxieme colonne ;
        determination des coordonnees X, Y de la derniere
            entite selectionnee qui est placee sur
            sur la deuxieme colonne ;
        placement des figures eventuelles ;
        placement des notes eventuelles ;
        creation d'une page ;
        gestion de la chaine des entites
      fin
    fin
  fin
fin
```

- Spécification des variables :

ECARTMOYEN	: écart moyen entre entités
ECARTMINIMUM	: écart minimum entre entités
ECARTMAXIMUM	: écart maximum entre entités
MARGESUPERIEURE	: marge supérieure d'une page du document
MARGEINFERIEURE	: marge inférieure du document
LONGUEURDISPONIBLE	: place restante sur la page
LARGPOIPA	: longueur d'une feuille du document
LONGPOIPA	: longueur d'une feuille du document
SAUTDEPAGE	: booléen indiquant si l'entité doit sauter de page
IERNOTE	: booléen indiquant si c'est la première note
MPIERFIG	: pointeur vers la première figure
TEST	: booléen indiquant si il y a encore des entités
SAUT	: booléen indiquant si il y a eu déjà un saut
SOMME	: variable de sommation de longueur des entités constituant les éléments d'identification
ECARTOPTIMAL	: facteur d'élasticité
COMPTEURINTER	: compteur intermédiaire du nombre d'entités sur la page
NBENTITEPAGE	: compteur du nombre d'entités sur la page
LONGUEURFIGURE	: place réservée aux figures
LONGUEURNOTE	: place réservée aux notes
RESTENOTE	: variable de mémorisation de la place occupée par les notes
ADRPREMIERENOTE	: pointeur vers la première note
MPDERNIERE	: pointeur vers la dernière entité sélectionnée pour la page
PTENTITE	: pointeur vers l'entité créée
MEMORYENTITE	: pointeur vers l'avant - dernière entité sélectionnée pour la page
PTPAGEENTITE	: pointeur vers la page créée
MEMOIREENTITE	: pointeur vers l'entité précédant la note ou la figure

2.3.2.1. PROCEDURE RESERVATIONNOTE

- Entrees :

LONGUEURNOTE	: place réservée sur la page pour les notes
ECARTMINIMUM	: écart minimum entre entités
ADRPREMIERENOTE	: pointeur vers la première note
ENTITECOURANTE	: pointeur vers l'entité courante
MEMOIREENTITE	: pointeur vers l'entité précédant la ou les note(s)
MPADRNOTE	: pointeur intermédiaire vers une note
IERNOTE	: booléen indiquant la première occurrence d'une note
LONGUEURDISPONIBLE	: place restante disponible sur la page

- Sorties :

idem sauf ECARTMINIMUM
Toutes ces variables sont mises à jour et ENTITECOURANTE devient le pointeur suivant la dernière note

- Effet :

Cette procédure a pour but de réserver de la place pour la note sur la page courante dans le cas du double colonnage.
La logique de cette procédure est la suivante : toutes les entités sont chaînées : c'est ce que nous appelons chaînage classique ; ces entités sont toutes les composantes de l'arbre de formatage. A chaque rencontre d'une entité note, la procédure la place en considérant pour celle-ci un espace entre les autres entités ; espace différent suivant que c'est la première note ou non. Elle crée ensuite une chaîne qui ne reprend que les notes et détache celle-ci de la chaîne classique.
La réservation tenant compte du double colonnage afin que la note ne chevauche pas les autres entités

- Pré-conditions :

(
(((ADRPRMIERENOTE = NIL) ∧ (IERNOTE = TRUE) ∧ (MPADRNOTE = NIL)) ∨
((ADPREMIERENOTE = NIL) ∧ (IERNOTE = FALSE) ∧ (MPADRNOTE <> NIL)))
(MEMOIREENTITE <> note) ∧
(0 <= LONGUEURNOTE <= LONGUEURFEUILLE) ∧
(largeur de l'entité note + ECARTMINIMUM <= LONGUEURDISPONIBLE
<= LONGUEURFEUILLE) ∧
(chaînage de toutes les entités classique et des notes éventuelles)
)

- Post-conditions :

(
(mise à jour de la chaîne classique des entités,
MEMOIREENTITE contenant l'adresse de l'entité suivant la note) ∧
(mise à jour de la chaîne des notes)
(((IERNOTE = FALSE) ∧ (MPADRNOTE = ENTITECOURANTE))
((IERNOTE = TRUE) ∧ (MPADRNOTE = ENTITECOURANTE) ∧
(ADRPREMIERENOTE = ENTITECOURANTE))) ∧
((0 <= LONGUEURDISPONIBLE <= LONGUEURFEUILLE - (2 * LONGUEURNOTE -
6 * ECARTMINIMUM)) ∧ ((IERNOTE = TRUE) ∨ (IERENOTE = FALSE)))
(((LONGUEURNOTE + 2 * ECARTMINIMUM) <= LONGUEURNOTE <= LONGUEURFEUILLE) ∧
(IERENOTE = FALSE)) ∨
(((LONGUEURNOTE + 6 * ECARTMINIMUM) <= LONGUEURNOTE <= LONGUEURFEUILLE)
(IERENOTE = TRUE))) ∧
(aucun chevauchement ni horizontal ni vertical de la note avec les
autres entités)
)

- Procédure appelante :

DEUXCOLO

- Procédure appelée :

rien

- Algorithme logique :

```
début
  si c'est la première note
    début
      tant qu'il y a des notes
        début
          incrémentation de la place réservée aux notes ;
          décrémentation de la place restante disponible pour
              les autres entités ;
          création de la chaîne des notes ;
          mise à jour de la chaîne classique des entités ;
          sélection de la note suivante
        fin
    fin
  sinon
    début
      tant qu'il y a des notes
        début
          mise à jour de la chaîne des notes ;
          incrémentation de la place réservée pour les notes ;
          décrémentation de la place restante disponible pour
              les autres entités ;
          mise à jour de la chaîne classique des entités ;
          sélection de la note suivante
        fin
    fin
fin
```

- Spécification des variables

RIEN

2.3.2.2. PROCEDURE RESERVATIONFIGURE

- Entrées :

LONGUEURFIGURE	: place réservée sur la page pour les figures
ECARTMINIMUM	: écart minimum entre entités
ADRIERFIGURE	: pointeur vers la première figure
ENTITECOURANTE	: pointeur vers l'entité courante
MEMOIREENTITE	: pointeur vers l'entité précédant la ou les figure(s)
MPADRFIG	: pointeur intermédiaire vers une figure
IERFIG	: booléen indiquant la première occurrence d'une figure
LONGUEURDISPONIBLE	: place restante disponible sur la page

- Sorties :

idem sauf ECARTMINIMUM
Toutes ces variables sont mises à jour et ENTITECOURANTE devient le pointeur suivant la dernière figure

- Effet :

Cette procédure a pour but de réserver de la place pour la figure sur la page courante dans le cas du double colonnage.
La logique de cette procédure est la suivante : toutes les entités sont chaînées : c'est ce que nous appelons chaînage classique ; ces entités sont toutes les composantes de l'arbre de formatage. A chaque rencontre d'une entité figure, la procédure réserve la place pour celle-ci en considérant un espace entre les autres entités ; espace identique suivant que c'est la première figure ou non. Elle crée ensuite une chaîne qui ne reprend que les figures et détache celle-ci de la chaîne classique.

- Pré-conditions :

(
(((ADRIREFIG = NIL) ∧
(((IERFIGURE = TRUE) ∧(ADRFIG = NIL))∨
((IERFIGURE = FALSE) ∧(ADRFIG <> NIL))))
(MEMOIREENTITE <> figure)∧
(0 <= LONGUEURFIGURE <= LONGUEURFEUILLE) ∧
(largeur de l'entité figure + ECARTMINIMUM <= LONGUEURDISPONIBLE
<= LONGUEURFEUILLE)∧
(chaînage de toutes les entités classiques et des figures éventuelles)
)

- Post-conditions :

(
(mise à jour de la chaîne classique des entités,
MEMOIREENTITE contenant l'adresse de l'entité suivant la figure) ∧
(mise à jour de la chaîne des figures) ∧
(((IERFIGURE = FALSE) ∧ (MPADRFIGURE = ENTITECOURANTE))∨
(((IERFIG = FALSE) ∧(ADRIERFIG = ENTITECOURANTE) ∧
(ADRFIG = ENTITECOURANTE)))∧
((0 <= LONGUEURDISPONIBLE <= LONGUEURFEUILLE - (LONGUEURFIGURE -
2 * ECARTMINIMUM)) ∧(IERFIG = TRUE) ∨(IERFIG = FALSE))
(((LONGUEURFIGURE + ECARTMINIMUM) <= LONGUEURFIGURE <= LONGUEURFEUILLE)
(IERFIG) ∧
(aucun chevauchement ni horizontal ni vertical de la figure avec les autres entités)
)

- Procédure appelante :

DEUXCOLO

- Procédure appelée :

rien

- Algorithme logique :

```
début
  si c'est la première figure
    début
      tant qu'il y a des figures
        début
          incrémentation de la place réservée aux figures ;
          décrémentation de la place restante disponible pour
              les autres entités ;
          création de la chaîne des figures ;
          mise à jour de la chaîne classique des entités ;
          sélection de la figure suivante
        fin
    fin
  sinon
    début
      tant qu'il y a des figures
        début
          mise à jour de la chaîne des figures ;
          incrémentation de la place réservée pour les figures
          décrémentation de la place restante disponible pour
              les autres entités ;
          mise à jour de la chaîne classique des entités ;
          sélection de la figure suivante
        fin
    fin
fin
```

- Spécification des variables :

rien

2.3.2.3. PROCEDURE PLACEMENTNOTE

- Entrées :

LONGUEURNOTE : place allouée sur la page pour les notes
ECARTMINIMUM : écart minimum entre entités
ADRPREMIERENOTE : pointeur vers la première note
PDOCUMENT : pointeur vers le document

- Sorties :

ADRPREMIERENOTE : idem
PDOCUMENT : idem

- Effet :

Cette procédure, connaissant l'adresse de la
première note et la place réservée pour
celle-ci arrange les notes en fin de page
et crée une séparation entre la ou les note(s) et
les autres entités

- Pré-conditions :

(
(LONGUEURFEUILLE > LONGUEURNOTE > 0) ∧
(ADRPREMIERENOTE <> NIL) ∧
(toutes les notes sont chainées)
)

- Post-conditions :

(
(LONGUEURNOTE = 0) ∧
(chaque note a sa coordonnée Y et sa coordonnée X
déterminée en fonction de la page)∧
(toutes les notes sont chainées) ∧
(écart entre les entités de type note = ECARTMINIMUM)∧
(toutes les notes sont placées) ∧
(les notes ne chevauchent aucune autre entité) ∧
(les notes sont placées par ordre décroissant de leur
référence)
)

- Procédure appelante :

DEUXCOLO

- Procédures appelées :

EXAMEN
CREERECORD
GERERRECORD

- Algorithme logique :

```
début
  création des records pour construire la séparation entre
  les notes et les autres entités ;
  gestion de l'entité créée ;
  placement des coordonnées X, Y de la première note ;
  sélection de la note suivante ;
  tant qu'il y a des notes
    début
      placement des coordonnées X, Y de la note sélectionnée ;
      décrémentation de la place allouée à la note ;
      sélection de la note suivante
    fin
fin
```

- Spécification des variables :

PENTITE : pointeur vers l'entité de séparation créée
PLIGNE : pointeur vers la ligne de séparation créée
PBOUTLI : pointeur vers le boutli de séparation crée
INDICENOTE : pointeur vers une note
INTERMEDIAIRE : pointeur de mémorisation d'adresse d'une note
POSITIONX : variable entière indiquant la coordonnée X
de l'entité séparation par rapport à la page POSITIONY
: idem mais coordonnée Y
LARGMOT : variable entière contenant la largeur de
la ligne de séparation
I : variable entière d'indice de boucle
LARGEUR : paramètre pour la procédure EXAMEN
HAUTEUR : idem
CARA : caractère "-" constituant la séparation
SORTE : variable déterminant le type d'élément
typographique

2.3.2.4. PROCEDURE PLACEMENTFIGURE

- Entrees :

ADRIEREFIGURE : pointeur vers la premiere figure
LONGUEURFIGURE : place reservee aux figures
ECARTMINIMUM : espace minimum entre entites
LONGUEURNOTE : place reservee aux notes

- sortie :

ADRIEREFIGURE : pointeur vers la premiere figure

- Effet :

Cette procedure determine les coordonnees Y des figures sur la page courante. L'entite figure est placee en fin de page apres les autres entites mais precedant les notes

- Pre-conditions :

(
(0 <= LONGUEURFIGURE <= (LONGUEURFEUILLE - LONGUEURNOTE)) ∧
(LONGUEURNOTE >= MARGEINFERIEURE) ∧
(les entites figures sont chainees)
)

- Post-conditions :

(
(chaque entite figure a sa coordonnee Y determinee de telle maniere que : soit y1 la coordonnee de l'entite 1, y2 la coordonnee de l'entite 2, et l'entite 1 precedant l'entite 2, la relation : y2 = y1 - longueur entite 1 - ecart minimum est toujours verifiee) ∧
(chainage des figures specifiques a la page) ∧
(LONGUEURFIGURE = 0) ∧
)

- Procedure appelante : DEUXCOLO

- Procedure appelee : rien

- Algorithme logique :

```
debut
  tant qu'il y a des figures
    debut
      determination de la coordonnee Y de la figure ;
      mise a jour de la place reservee aux figures ;
      selection de la figure suivante
    fin
fin
```

- Specification des variables :

INDICEFIGURE : pointeur intermediaire vers une figure

2.3.2.5. PROCEDURE CREERPAGE

- Entrées :

ADRPREMIEREPAGE	: pointeur vers la première page
ADRDERNIEREPAGECREEE	: pointeur vers la dernière page
IEREPAGE	: booléen indiquant si une page a déjà été créée pour le document
PTPAGE	: pointeur vers la page créée
ADRPREMIERENOTE	: pointeur vers la première note
MEMENTITE	: pointeur vers la première entité de la page
ADRIEREFIGURE	: pointeur vers la première figure

- Sorties :

ADRPREMIEREPAGE	: idem
ADRDERNIEREPAGECREEE	: idem
IEREPAGE	: idem
PTPAGE	: idem

- Effet :

Cette procédure crée une page et met à jour les pointeurs de la page et de la chaine des pages : elle crée la chaine des pages et relie la page courante à la précédante, établit le lien entre la page créée et ses entités, et relie la dernière entité de la page à la première note bas de page si elle existe et la première figure si elle existe à la dernière entité ou à la dernière note bas de page si elle existe

- Pré-conditions :

```
(
((IERENOTE) ∧
 (ADRPREMIEREPAGE = NIL) ∧ (ADRDERNIEREPAGECREE = NIL)) ∧
 (MEMENTITE <> NIL) ∧
 (ADRPREMIERENOTE ∧ ADRIEREFIGURE)
 )
```

- Post-conditions :

```
((IERNOTE = FALSE) ∧ (ADRPREMIEREPAGE = NIL)
 (ADRDERNIEREPAGECREEE = PTPAGE)) ∧
(toutes les entités spécifiques à une page sont chainées) ∧
(toutes les notes bas de page spécifiques à une page sont
 chainées et la première est reliée à la dernière entité) ∧
((la première figure est chainée à la première note) ∧
 (elle est chainée ala dernière entité de la page))
 )
```

- Procédure appelante :

DEUXCOLO

- Procédure appelée :

rien

- Algorithme logique :

```
début
  création d'une page ;
  si c'est la première page du document
    création de la chaine des pages ;
  sinon
    mise à jour de la chaine des pages
  mise à jour des éléments de la page dont la filiation
        page - entité ;
  tant qu'il reste des entités pour cette page
    début
      mémorisation de l'entité ;
      sélection de l'entité suivante
    fin
  lien de la dernière entité et de la première note
  si il existe au moins une note
    début
      tant qu'il y a des notes
        début
          mémorisation de la note ;
          sélection de la note suivante
        fin
      lien de la première figure et de la dernière note ;
    fin
  sinon
    lien de la première figure et de la dernière entité
fin
```

- Spécification des variables :

MPMEM	: pointeur intermédiaire vers une entité
MPMEMOIRE	: pointeur de mémorisation de la dernière entité de la page
MPTIG	: pointeur intermédiaire vers une note
MPMEMTIG	: pointeur vers la dernière note de la page

2.3.2.6. PROCEDURE COLONNAGE

- Entrees :

LONGUEURDISPONIBLE : place restante sur la page
PTPAGEENTITE : pointeur vers l'entité créée
MEMOIREENTITE : pointeur vers l'entité précédant celle qui chevauche
LONGUEURNOTE : place réservée au(x) note(s)
LONGUEURFIGURE : place réservée au(x) figure(s)

- Sorties :

MEMOIREENTITE : idem
PTPAGEENTITE : idem

- Effet

Cette procédure, recevant les paramètres de mise en page et le pointeur vers l'entité chevauchante, gère le passage partiel de l'entité de la première à la deuxième colonne ; elle détermine les coordonnées X, Y de l'entité cassée et de l'entité créée et réorganise le placement des lignes et des bouts de ligne au sein de ces deux entités ; la procédure gère également la chaine des entités sélectionnées pour la page courante et renvoie le pointeur de l'entité créée et de l'entité précédante

- Pré-conditions :

(
(MEMENTITE <> NIL) ∧ (PTPAGEENTITE = NIL) ∧ (MEMORYENTITE <> NIL) ∧
(MPCOMPTEUR > 1) ∧
(toutes les entités sélectionnées pour la page courante sont chainées)
)

- Post-conditions :

(
((les lignes au sein de l'entité créée et cassée sont rangées de telle manière que : soit y1, y2 respectivement les coordonnées de ligne 1 et ligne 2, et ligne 1 précédant ligne 2, la relation : y2 = y1 - largeur ligne 1 - interligne est toujours vérifiée) ∧ (PTENTITE <> NIL)) ∧
((mêmes conditions pour les bouts de ligne DECR étant le facteur de réajustement pour l'entité cassée)) ∧
((CLIGNE > 1) ∧ (MEMLIGNE <> NIL))
(((FANION = FALSE) ∧ (MEMORYENTITE = ENTITECOURANTE) ∧ (MEMPENTI <> NIL)) ∨
((FANION = TRUE) ∧ (MEMORYENTITE = PTENTITE) ∧ (MEMPENTI <> NIL))) ∧
(la coordonnée Y de la dernière entité placée dans la première colonne est telle que : y = placement de l'entité précédante + longueur des notes et des figures) ∧
(les entités spécifiques à la page sont chainées) ∧
(la place disponible restante, après avoir placé l'avant-dernière entité sélectionnée pour la page courante indépendamment de la colonne < longueur de l'entité courante < longueur d'une feuille du document)
)

- Procédure appelante :

POSITIONY1

- Procédure appelée :

rien

- Algorithme logique :

```
début
  tant que (nombre d'entités sélectionnées - 1) > 0
    début
      mémorisation de l'entité ;
      sélection de l'entité suivante
    fin
  si la dernière entité placée <> nil
    début
      initialisation de la place disponible ;
      mémorisation de la première ligne ;
      calcul de la place restante sur la première colonne ;
      si il ne reste pas assez de place, report de l'entité à la
          deuxième colonne ;
      sinon
        début
          réinitialisation de la place disponible ;
          mise à jour de l'entité courante
        fin
    fin
  tant qu'il reste de la place
    début
      détermination des coordonnées Y des lignes ;
      incrémentation du compteur de ligne ;
      mémorisation de la ligne ;
      sélection ligne suivante
    fin
  mise à jour de la chaine des lignes ;
  création d'une nouvelle entité ;
  mise à jour des éléments de l'entité créée ;
  réajustement des coordonnées Y des lignes et des bouts de ligne
      au sein de l'entité créée ;
  mise à jour de la chaine des entités
fin
```

- Spécification des variables :

MEMOIRELIGNE	: pointeur vers la dernière ligne de l'entité cassée
PTENTITE	: pointeur vers l'entité créée
MEMLIGNE	: pointeur vers la première ligne de l'entité créée
LIGNE	: pointeur de mémorisation d'une ligne
MEMPENTI	: pointeur de mémorisation d'une entité
BOUTLI	: pointeur vers un bout de ligne
TEST	: place restante disponible pour placer les lignes de l'entité chevauchante
DECR	: facteur de réajustement des lignes et des bouts de ligne au sein de l'entité créée
CLIGNE	: compteur du nombre de lignes
TAILLE	: place restante disponible lorsque l'avant-dernière entité a été placée
FANION	: booléen indiquant si l'entité chevauche sur les colonnes ou sur les pages

2.3.2.7 PROCEDURE SAUTPAGE

- Entrées :

MEMOIREENTITE : pointeur vers l'entité précédant l'entité qui saute de page
LONGUEURDISPONIBLE : place restante sur la page
LONGUEURNOTE : place réservée aux notes
LONGUEURFIGURE : place réservée pour les figures
PTPAGEENTITE : pointeur vers l'entité créée

- Sorties :

MEMOIREENTITE : idem
PTPAGEENTITE : idem

- Effet :

Cette procédure a pour but de gérer le remplissage de la page dans le cas du passage d'une entité de la première à la deuxième colonne ; elle détermine les coordonnées X, Y des entités de la première colonne, gère la cassure de l'entité chevauchante sur les deux colonnes, crée une nouvelle entité, gère la chaine des pointeurs des entités et met à jour les coordonnées X, Y des lignes et des bouts de ligne au sein de l'entité cassée et créée

- Pré-conditions :

(
((MEMOIREENTITE <> NIL) ∧ (PTPAGEENTITE = NIL)) ∧
(longueur de MEMOIREENTITE > (LONGUEURDISPONIBLE / 2 - (LONGUEURFIGURE + LONGUEURNOTE))) ∧
(les entités spécifiques à la page sont chainées)
)

- Post-conditions :

(
(PTPAGEENTITE <> NIL) ∧
(l'espace entre l'entité cassée et l'entité précédante est l'ECARTMOYEN) ∧
(les coordonnées Y des lignes et des bouts de ligne au sein de l'entité cassée et créée sont telles que : soit y1 la coordonnée de la ligne 1, y2 la coordonnée de la ligne 2, et la ligne 1 précédant la ligne 2, la relation : y2 = y1 - longueur ligne 1 - interligne est toujours vérifiée : de même pour les bouts de ligne) ∧
(si x1 désigne la coordonnée X des entités de la première colonne et x2 la coordonnée X des entités de la deuxième colonne : x1 = 0 et x2 = 945) ∧
(les entités spécifiques à la page sont chainées)
)

- Procédure appelante :

DEUXCOLO

- Procédure appelée :

rien

- Algorithme logique :

```
debut
  tant qu'il reste de la place
    debut
      détermination de la coordonnée Y de la ligne ;
      sélection de la ligne suivante
    fin
  détermination des coordonnées X, Y de l'entité ;
  création d'une nouvelle entité ;
  mise à jour de la chaine des pointeurs et des éléments
       de l'entité cassée et créée ;
  tant qu'il reste des lignes
    debut
      mise à jour des lignes et des bouts de ligne ;
      sélection ligne et bout de ligne suivants
    fin
fin
```

- Spécification des variables :

PLIGNE : pointeur intermédiaire vers une ligne
PMEMOIRE : pointeur vers la dernière ligne de la dernière entité de la première colonne
MEMLIGNE : pointeur de la première ligne de la première entité de la deuxième colonne
BOUTLI : pointeur intermédiaire vers un bout de ligne
SOMME : variable entière déterminant la longueur de l'entité cassée
TEST : place restant disponible sur la première colonne de la page courante
DECR : facteur de réajustement des lignes et des bouts de ligne au sein de l'entité cassée et créée
CLIGNE : compteur du nombre de lignes de la dernière entité placée sur la première colonne

2.3.2.8. PROCEDURE SEULESURPAGE

- Entrées :

ADRPREMIEREPAGE	: pointeur vers la première page
ADRDERNIEREPAGECREE	: pointeur vers la dernière page du document
MEMENTITE	: pointeur vers la première entité
PDOCUMENT	: pointeur vers le document
IEREPAGE	: booléen indiquant si le document contient une ou plusieurs pages

- Sorties :

ADRDERNIEREPAGECREE	: idem
ADRPREMIEEPAGE	: idem
MEMENTITE	: idem
IEREPAGE	: idem

- Effet :

Cette procedure a pour but de déterminer les coordonnées Y des entités constituant les éléments d'identification (titre, auteur, date, résumé) de telle manière que ces entités soient seules sur la première page du document et centrées.
Elle gère également les notes éventuelles

- Pré-conditions :

(
(toutes les entités sont chainées) ∧
((ADRPREMIEREPAGE = NIL) ∧ (ADRDERNIEREPAGECREEE = NIL)
(MEMENTITE <> NIL)) ∧
((IERPAGE = TRUE)∧(PDOCUMENT <> NIL)) ∧
(il existe au moins une entête)
)

- Post-conditions :

(
(les entités appartenant à la première page du document sont chainées) ∧
((IEREPAGE = FALSE) ∧(ADRPREMIERPAGE <> NIL) ∧
(ADRDERNIEREPAGECREE <> NIL) ∧ (MEMENTITE = entête))
(les coordonnées Y des entités sélectionnées sont telles que : soit y1 la coordonnée Y de l'entité 1, y2 la coordonnée Y de l'entité 1, et l'entité 1 précédant l'entité 2, la relation : y2 = y1 - longueur entité 1 - DEBUT est toujours vérifiée ; DEBUT étant le facteur de centrage) ∧
((NBENTITEPAGE = 1) ∧ (tous les paramètres de la gestion de la mise en page sont déterminés : MARGEINFERIEURE, MARGESUPERIEURE, LONGUEURNOTE, LONGUEURFIGURE, LONGUEURDISPONIBLE, ECARTMOYEN, ECATMINIMUM, ECARTMAXIMUM)
)

- Procédure appelante :

MISEENPAGE

- Procedures appelées :

URESERVATIONNOTE
UPLACEMENTNOTE
UCREERPAGE

- Algorithme logique :

```
début
  initialisation des paramètres de mise en page ;
  tant que le type d'entité est différent de entête
    début
      si le type d'entité est une note, réservation de la place
      sinon
        début
          décrémentation de la place disponible ;
          mémorisation de l'entité ;
          sélection de l'entité suivante
        fin
      gestion de la chaine des entités ;
      initialisation des paramètres de centrage ;
      placement de la première entité ;
      tant qu'il reste des entités
        début
          détermination des coordonnées Y de l'entité ;
          sélection de l'entité suivante
        fin
      si il exite des notes, détermination de leur coordonnée Y ;
      création d'une page ;
      mise à jour de l'entité suivante à traiter
  fin
fin
```

- Spécification des variables :

ENTITECOURANTE	: pointeur vers une entité
ENTITE	: pointeur intermédiaire vers une entité
RES	: pointeur vers la première entité de la page suivante
ADRPREMIERENOTE	: ponteur vers la première note
MEMOIREENTITE	: pointeur vers la dernière entité de la page
ADRIERFIGURE	: pointeur vers la première figure
MPADRNOTE	: pointeur vers la dernière note
PTPAGE	: pointeur vers la page créée
PREC	: pointeur vers l'entité précédant la courante
DEBUT	: coordonnée Y de la première entité
LONGUEURDISPONIBLE	: place restante sur la page
LONGUEURNOTE	: espace réservé aux notes
ECARTMINIMUM	: écart minimum entre entités
ECARTMOYEN	: écart moyen entre entités
LARGEUR	: espace réservé pour les éléments d'identification
NBENTITEPAGE	: nombre d'entité sélectionnées pour la page
COMPTEURNOTE	: nombre de notes sélectionnées pour la page
LONGPOIPA	: longueur d'une feuille
IEIRENOTE	: booleen indiquant la première occurrence d'une note

2.3.2.9. PROCEDURE POSITIONY1

- Entrées :

LONGUEURDISPONIBLE	: place restante sur la page
ENTITECOURANTE	: pointeur vers l'entite courante
MEMENTITE	: pointeur vers la premiere entite de la page
PTENTITE	: pointeur vers l'entite creee
LONGPOIPA	: longueur d'une feuille du document
MARGESUPERIEURE	: marge superieure d'une page
MARGEINFERIEURE	: marge inferieure d'un page
ECARTMINIMUM	: ecart minimum entre entites
LONGUEURNOTE	: espace reserve aux notes
ECARTMAXIMUM	: ecart maximum entre entites
LONGUEURFIGURE	: espace reserve aux figures
SAUTDEPAGE	: booleen indiquant si une entite a deja saute de page

- Sorties : MEMORYENTITE, ENTITECOURANTE, MEMENTITE, PTENTITE : idem

- Effet :

Cette procedure, à partir des parametres de mise en page et du pointeur vers la premiere entite, determine les coordonnees X, Y des entites selectionnees pour la page courante. Elle organise le placement des entites en double colonne et renvoie le pointeur de la premiere entite de la page courante et le pointeur de la derniere entite placee sur la page ou destinee à etre placee sur la page suivante

- Pre-conditions :

(
((MEMENTITE <> NIL) ∧ (PTENTITE <> NIL) ∧ (ENTITECOURANTE <> NIL))∧
(la chaine des entites est mise à jour de telle maniere que les entites destinees à etre placees sur la page en double colonne remplissent entierement ou partiellement la page : somme des longueurs des entites < LONGUEURDISPONIBLE)∧
(pointeur suivant de la derniere entite selectionnee = NIL)∧
(SAUTDEPAGE = FALSE)∧
(MEMORYENTITE = NIL)
)

- Post-conditions :

(
(((LONGUEURDISPONIBLE < 0) ∧(SAUTDEPAGE = TRUE)
(MEMORYENTITE = avant-derniere entite placee))∨
((LONGUEURDISPONIBLE > 0) ∧(SAUTDEPAGE = FALSE)∧
(MEMORYENTITE = derniere entite placee)))
(les coordonnees Y des entites sont determinees de telle maniere que : soit y1, y2 respectivement les coordonnees Y de entite 1 et entite 2, et entite 1 precedant l'entite 2, la relation : y2 = y1 - longueur entite 1 - ecartmoyen est toujours verifiee)∧
(((TEST = TRUE) ∧(LONGUEURDISPONIBLE))∨
((TEST = FALSE) ∧(LONGUEURDISPONIBLE)))
(les coordonnees X des entites sont determinees de telle maniere que : soit x1, x2 les coordonnees respectivement de entite 1 et entite 2, entite 1 etant sur la premiere colonne et entite 2 sur la deuxieme, la relation : x2 = 945 x1 = o est toujours verifiee)∧
(aucun chevauchement d'entite)
)

- Procedure appelante :

DEUXCOLO

- Procedure appelee :

COLONNAGE

- Algorithme logique :

```
debut
  initialisation des parametres de mise en page pour le double
      colonnage ;
  tant qu'il reste des entites
    debut
      tant qu'il reste de la place
        debut
          determination des coordonnee X, Y de l'entite ;
          memorisation de l'entite placee ;
          gestion de la chaine des entites ;
          selection de l'entite suivante
        fin
  si la place disponible est devenue negative
    debut
      gestion du passage partiel de l'entite chevauchante de la
          premiere à la deuxieme colonne ;
      reinitialisation des parametres de mise en page ;
      si l'entite chevauchante est deja sur la deuxieme colonne
        debut
          initialisation des parametres pour le saut de l'entite
          rearangement de la derniere entite afin d'avoir
              un remplissage optimal de la page
        fin
    fin
  fin
fin
```

- Spécification des variables :

POSITION	: coordonnee X de l'entite
NPCOMPTEUR	: compteur du nombre d'entites placees
TEST	: booleen indiquant si il reste des entites à traiter ou non

2.4. PROCEDURE NUMEROTATION

- Entrées :

POCUMENT : pointeur vers le document
ADRPREMIEREPAGE : pointeur vers la première page

- Sortie :

ADRPREMIEREPAGE : pointeur vers la première page

- Effet :

Cette procédure, recevant le pointeur de la première page et celui du document, attribue un numéro séquentiel à chaque page . Cette numérotation est centrée soit au-dessus de la page, soit au-dessous de la page

- Pré-conditions :

(
ADRPREMIEREPAGE ∧ POCUMENT
)

- Post-conditions :

(
((SUPERIEURE) /\ (chaque page est dotée d'un numéro entier centré sur la page) /\(0 <numéro < 9999)) ∨
((INFERIEURE chaque page est dotée d'un numéro entier centré sur le bas la page) ∧(0 < numéro < 9999))
)

- Procédure appelante :

MISEENPAGE

- Procédures appelées :

CREERECORD
GERERECORD
EXAMEN

- Algorithme logique :

```
debut
  lecture de la première page ;
  tant qu'il y a des pages
    debut
      sélectionner la première page ;
      création d'un record entité,ligne et bout de ligne ;
      détermination du numéro et de ses dimensions ;
      gestion des records créés par remplissage de leurs
          champs et  mise à jour des pointeurs ;
      si SUPERIEURE, chainage de l'entité créée à la
          première entité de la page ;
      sinon chainage de l'entité créée à la dernière
          entité de la page
    fin
  lecture de page suivante
fin
```

- Spécification des variables :

PAGECOURANTE	: pointeur vers une page
PTENTITE	: pointeur vers une entité
PTENTITECOURANTE	: pointeur vers l'entité courante
DERNIEREENTITE	: pointeur vers la dernière entité de la page
PTENTITE	: pointeur vers l'entité créée
PLIGNE	: pointeur vers la ligne créée
PBOUTLI	: pointeur vers le bout de ligne crée
POSITIONX	: coordonnée X du numéro par rapport à la page
DIVISEUR	: variable entière contenant le résultat de la division entière du nombre de pages par 100
NB	: variable entière contenant le nombre de chiffres que contient le numéro
NUMPAGE	: variable entière contenant le nombre de pages
CHIFFRECARA	: variable entière contenant le reste de la division du nombre de pages par un multiple de 10
LARGMOT	: variable entière contenant la largeur du mot à créer : largeur des caractères formant le numéro + largeur des "-" et des blancs
I	: variable entière d'indice de boucle
LARGEUR	: paramètre pour la procedure EXAMEN
MEMPOSITION	: variable entière intermédiaire de position du mot
CARA	: caractère "chiffre" analysé pour connaître ses dimensions
SORTE	: variable indiquant le type d'élement typographique

2.5. PROCEDURE TABLEMATIERE

- Entrées :

ARBFOR : pointeur vers l'arbre de formatage
ARBSTRUC : pointeur vers l'arbre de structure
IERTAB : pointeur vers l'arbre de formatage

- Sortie :

IERTAB : pointeur vers l'arbre de formatage

- Effet :

Cette procédure construit une table des matières à partir des entêtes de chapitre.

- Pré-conditions :

(
ARBFOR ∧ IERTAB ∧ ARBSTRUC
)

- Post-condition :

(
table des matières créée et liée au dernier record de l'arbre de formatage
)

- Procédure appelante :

ABRFORMATAGE

- Procédures appelées :

PROCENT
TRITABLE
TRITABLE

- Algorithme logique :

```
debut
  initialisation des variables ;
  création d'une première page ;
  construction de l'entête de la table des matieres ;
  liaison de l'entête et de la page créée ;
  construction de la table à partir des entêtes de chapitre ;
  liaison de la première page au dernier record de l'arbre de formatage
fin
```

- Spécification des variables :

CPTENT	: compteur du nombre d'entêtes
MRS	: nombre de millimètres pour la marge inferieure
VAL	: entier
I	: compteur
LARGEURBLANC	: nombre de millimetres pour le caractere blanc
HAUT	: hauteur d'un caractère
LARG	: largeur d'un caractère
HAUTEURMOT	: hauteur d'un bout de ligne
LARGEURMOT	: largeur d'un bout de ligne
LONGDISPONIBLE	: nombre de millimètres disponibles en largeur
NBNIV	: nombre de niveaux d'entête à prendre en consideration
VALI	: compteur intermédiaire du nombre d'entêtes
INTERTABS	: nombre de millimètres entre deux lignes
MRDTAB	: nombre de millimètres pour la marge droite
MRGTAB	: nombre de millimètres pour la marge droite
POLTAB	: type de police
NBPPAGE	: nombre de page du document formate
LONGPOIPA	: nombre de millimètres en longueur de la feuille
LARGPOIPA	: nombre de millimètres en largeur de la feuille
FOTAB	: type de fonte
SOUTAB	: type de soulignement
MEMPPAGE	: pointeur vers une page
PENTITE	: pointeur vers l'entité courante
PLIGNE	: pointeur vers la ligne courante
PBOUTLI	: pointeur vers le bout de ligne courant
VPBOUTLI	: mémorisation d'un pointeur vers un bout de ligne
MEMENTITE	: mémorisation d'un pointeur vers une entité
SORTE	: type de record de l'arbre de structure
SEULE	: /sans utilité ici
IERLIGNE	: /sans utilité ici
TROUVER	: /sans utilité ici

2.5.1. PROCEDURE GESTIONMATIERE

- Entrée :

ARBSTRUC : pointeur vers l'arbre de structure

- Sortie :

rien

- Effet :

Cette procédure a comme but , si le record courant de l'arbre de structure est de type entete et de niveau inferieur à une constante, d'inserer cette entête dans la table des matieres avec son numéro de page.
Cette procedure effectue le lien entre l'élement crée et les elements précédant de la table, tout en effectuant une mise en page.

- Pre-condition :

(
ARBSTRUC
)

- Post-conditions :

(
((ARBSTRUC^.TYPEUILLE = ENTETE) ∧(ARBSTRUC^.NIVEAUCHAPITRE = VALNIV)
∧(record de type entite crée)
∧(valeur assignée pour le type de soulignement, le type de fonte, le type de police, le saut de ligne de chaque bouts de ligne de l'entité créée)
∧(((il y a de la place sur la page courante)
∧(entite positionnée sur la page courante))
∨(il n'y a pas de place sur la page courante)
∧(une page créée)
∧(entite positionnée sur la page créée)
)
∧(un bout de ligne crée avec le numero de l'entete)
)

- Procédure appelante :

TABLEMATIERE

- Procedures appelées :

GERERRECORD
CREERRECORD
PROCENT

- Algorithme logique :

```
début
  si positionnement sur une entité de type entête dont le niveau
  est inférieure à une constante
    début
      création d'une entité;
      assignation d'une valeur définie à chaque bouts de ligne
       de l'entité ;
      si première entête, lien de l'entité avec la page
      sinon
        début
          si il y a de la place sur la page courante,
            liaison de l'entête avec la dernière entité créée
          sinon
            début
              création d'une page ;
              liaison de la page à la page précédente ;
              liaion de l'entête à la page
            fin
        fin
      création d'un bout de ligne comprenant le numéro de la page où
      se trouve l'entête dans le document ;
      liaison du bout de ligne créé au dernier bout de ligne de l'entité
    fin
fin
```

- Spécification des variables :

PMOT : mémorisation d'un pointeur vers un bout de ligne

2.5.2. PROCEDURE PROCENT

- Entrées :

CPTENT : numéro de l'entête recherchée
ARBFOR : pointeur vers l'arbre de formatage
VALI : numéro de l'entête courante

- Sorties :

VALI : numéro de l'entête courante mis à jour
NBPPAGE : numéro de la page de l'entête

- Effet :

Cette procédure parcourt l'arbre de formatage et rend le pointeur vers l'entité de type entête recherchée ainsi que le numéro de la page dans laquelle se trouve l'entête.

- Pré-conditions :

(
CPTENT ∧VALI ∧ ARBFOR
)

- Post-conditions :

(
(ARBFOR = pointeur vers l'entité de type entête recherchée)
∧(NBPPAGE = numéro de la page)
)

- Procédure appelante :

GESTIONMATIERE

- Procédure appelée :

rien

- Algorithme logique :

```
début
  tant que le numéro de l'entité pointée est <> du numéro
  de l'entité recherchée
    début
      si positionnment sur un record de type page
        début
          incrémentaion du numéro de page ;
          mémorisation de l'adresse de cette page
        fin
      si positionnement sur un record de type entité
        début
          si le type est 'entete', incrementation de son numéro ;
          si le pointeur vers le record suivant est nil
            début
                initialisation de l'adresse de parcours à celui de
                la page suivant celle mémorisée
            fin
        fin
    fin
fin
```

- Spécification des variables :

PAGE : mémorisation de l'adresse d'une page
ENTITE : mémorisation de l'adresse d'une entité

2.5.3. PROCEDURE TRITABLE

- Entrée :

ARBSTRUC : pointeur vers l'arbre de structure

- Sortie :

rien

- Effet :

Cette procédure crée la table des matières avec les records de type entete de l'arbre de structure et les relie à la première page de la table - page déjà créée.

- Pre-condition :

(
ARBSTRUC
)

- Post-condition :

(
table des matieres créée
)

- Procédure appelante :

TABLEMATIERE

- Procedures appelees :

GESTIONMATIERE
TRITABLE

- Algorithme logique :

début
parcours récursif de l'arbre de structure,
si un record est de type 'entete', création d'une entité pour la table des matieres
fin

- Spécification des variables :

rien

2.0.0.1 PROCEDURE EXAMEN

- Entrées :

TYCHAR : grandeur du caractère (dépend de la police)
TYFONT : type de fonte
CH : caractère à analyser

- Sorties :

TPHAUTEUR : hauteur du caractère
TPLARGEUR : largeur du caractère

- Effet :

Cette procédure, recevant un caractère, le type de fonte et de police, renvoit la hauteur et la largeur de ce caractère en nombre de points typographiques

- Preconditions :

```
(
(TYFONT ∧ (1 <= TYCHAR <= 20) ∧ CH)
)
```

- Post-conditions :

```
(
 TPLARGEUR ∧ TPHAUTEUR
)
```

- Procédures appelantes :

TRAITEMENTCLASSIQUE
TRIMOT
TRICHANGEMENT
TABLEMATIERE
NUMEROTATION
URESERVATIONNOTE

- Procédure appelée :

rien

- Algorithme logique :

```
début
  decodage du caractère en sa valeur ASCII ;
  détermination de la hauteur et de sa hauteur  en fonction
      de la police et de la fonte
fin
```

- Spécification des variables :

VAL : variable entière contenant la valeur ASCII du caractère

2.0.0.2. PROCEDURE BINAIREDECIMAL

- Entrée :

BYY : byte

- Sortie :

ENTIER : variable entiere

- Effet :

Cette procedure recevant un caractère contenu dans un byte le decode et rend sa valeur entière décimale

- Pré-condition

```
(
 BYY
)
```

- Post-conditions :

```
(
(((BYY[0] = 1) ∧ (ENTIER - 128 >= 0)) ∧ (BYY[0] = 0))∧
(((BYY[1] = 1) ∧ (ENTIER - 192 >= 0)) ∧ (BYY[1] = 0))∧
       suite cfr listing
)
```

- Procédure appelante :

TABLEAU

- Procédure appelée :

rien

- Algorithme logique :

```
début
  initialisation de la valeur entière du caractère ;
  examen de chaque bit et suivant sa valeur, incrémentation de
      la valeur entière du caractère
fin
```

- Spécification des variables :

rien

2.0.0.4. PROCEDURE GERERECORD

- Entrées :

GX	: position X de l'élément
GY	: position Y de l'élément
DELTAX	: largeur de l'élément
DELTAY	: hauteur de l'élément
NOMBREELEM	: nombre d'éléments
GSPACE	: largeur du ' '
GBLANC	: nombre de ' '
PTM	: pointeur vers l'élément
SORTE	: type de l'élément

- Sortie :

rien

- Effet :

Cette procédure à pour but de mettre à jour les éléments du record de type entité, ligne ou bout de ligne

- Pré-conditions :

(
GX ∧ GY ∧ DELTAX ∧ DELTAY ∧ NOMBREELEM ∧ GSPACE ∧ GBLANC ∧ PTM
∧ SORTE
)

- Post-condition :

(
le record pointé par PTM est mis à jour
)

- Procédures appelantes :

TRAITEMENTCHANGEMENT
TRAITEMENTCLASSIQUE
TABLEMATIERE
NUMEROTATION
URESERVATIONNOTE

- Procédure appelée :

rien

- Algorithme logique :

début
mise à jour des éléments du record
fin

- Spécification des variables :

rien

2.0.0.3. PROCEDURE CREERRECORD

- Entrées :

PO	: type de police
FO	: type de fonte
SOU	: type de soulignement
SORTE	: type de l'élément typographique
NBPOINTENTITE	: largeur de l'élément typographique

- Sortie :

PTM : pointeur vers le record de l'arbre de formatage créé

- Effet :

Cette procédure, recevant les caractéristiques de formatage, crée un élément typographique soit de type entité, ligne, ou bout de ligne qui contient ces caractéristiques et les coordonnées de placement et initialise les pointeurs

- Pré-conditions :

(
PO ∧ FO ∧ SOU ∧ SORTE ∧ NBPOINTENTITE
)

- Post-conditions :

(
(PTM <> NIL) ∧ (les éléments du record créé sont initialisés)
)

- Procédures appelantes :

TRAITEMENTCHANGEMENT
TRAITEMENTCLASSIQUE
TABLEMATIERE
NUMEROTATION
URESERVATIONNOTE

- Procédure appelée :

rien

- Algorithme logique :

début
création de l'élément typographique ;
initialisation des éléments du record
fin

- Spécification des variables :

rien

3. PROCEDURE FDDTRAITEMENT

- Entrée :

MEMPAGE : pointeur vers la première page de l'arbre de formatage

- Sortie :

IERTABRECORD : pointeur vers le premier bloc contenant les indications pour l'impression en format FDD

- Effet :

Cette procédure crée un ensemble de blocs chaînés contenant la description du document à imprimer dans le format FDD.

- Pre-condition :

(
MEMPAGE
)

- Post-condition :

(
IERTABRECORD
)

- Procédure appelante :

PROGRAMME PRINCIPAL

- Procédures appelées :

CONSTRDIRECTORY
CONSTRUCPARTDIRECTORY
PARCOURS
TRANSFORMEWORD
REMPLIRWORD

- Algorithme logique

```
debut
  construction du "HEADER" ;
  construction des "PARTS DIRECTORY"
  pour chaque page du document, construction du descripteur de page ;
  initialisation du nombre de pages dans le premier bloc ;
  initialisation des mots dans le premier bloc ;
fin
```

- Spécification des variables :

ADPAGE : compteur de pages
ZERO : constante de valeur 0 (utilité : impossibilité de passer en paramètre une valeur)
NBPAGE : nombre de pages du document
I : compteur
CPTPAGE : variable indiquant la page courante lors du parcours de l'arbre de formatage
COMPTEURTABRECORD : compteur du nombre de blocs créés
COMPTEURINF : compteur du nombre de "bytes" créés
TABRECORD : pointeur vers le bloc courant
TAB : pointeur vers un tableau contenant les indications que l'on n'a pu insérer directement dans les blocs
ADRFIRSTA : pointeur vers le premier élément du tableau
VALpage : pointeur vers le record courant indiquant la page à traiter de l'arbre de formatage
FIRSTA : booléen à true si positionnement sur le premier bloc
VL1 : premier "byte" d'un "word", en représentation binaire
VL2 : deuxième "byte" d'un "word" , en représentation binaire
VL11 : "word" décodé en binaire

3.1. PROCEDURE CONSTRDIRECTORY

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procédure construit le "DOCUMENT HEADER" du fichier FDD.
  (cfr page 3. du document "FORMATTED DOCUMENT DESCRIPTION" de R.D. HERSCH)

- Pré-condition :

  rien

- Post-conditions :

  (
  (création d'un bloc)
  ∧(remplissage des 64 premiers "words")
)

- Procédure appelante :

  FDDTRAITEMENT

- Procédures appelées :

  MISEAMOINSUN
  MISERECORD
  INCR

- Algorithme logique :

  début
  création d'un premier bloc ;
  initialisation des variables (compteur, pointeur premier record,..) ;
  remplissage des éléments du bloc ;
  fin

- Spécification des variables :

  VAR1 : premier "byte" d'un "word"
  VAR2 : second "byte" d'un "word"
  i : compteur
  vll3 : "word"
  x : jour du mois
  y : mois de l'année
  z : année

3.2. PROCEDURE CONSTRPARTDIRECTORY

- Entrée :

  rien

- Sortie :

  rien

- Effet :

  Cette procédure réserve la place pour la "PART DIRECTORY" d'une page
  (Cfr. page 3. du document "FORMATTED DOCUMENT DESCRITPION" de R.D. HERSCH)

- Pré-condition :

  rien

- Post-condition :

  (4 places "réservées" dans le bloc courant)

- Procédure appelante :

  FDDTRAITEMENT

- Procédure appelée :

  MISERECORD

- Algorithme logique :

  début
  remplissage de 4 "words" avec la valeur nul ;
  fin

- Spécification des variables :

  rien

3.3. PROCEDURE PARCOURS

- Entrées :

NBPAGE : nombre de pages du document
CPTPAGE : numéro de la page courante
MEMPAGE : pointeur vers le record de l'arbre de formatage de la page courante

- Sorties :

CPTPAGE : numéro de la page courante mis à jour
MEMPAGE : pointeur vers le record de l'arbre de formatage de la page courante mis à jour

- Effet :

Cette procédure parcourt récursivement l'arbre de formatage, pour chacune des pages on crée la "PRINTED page PART"
(cfr page 5. du document "FORMATTED DOCUMENT DESCRIPTION" de R.D. HERSCH)

- Pré-conditions

(
CPTPAGE ∧ MEMPAGE ∧ NBPAGE
)

- Post-conditions

(
CPTPAGE ∧ MEMPAGE ∧ NBPAGE ∧
(ensemble de blocs contenant les éléments des "PRINTED page PART")
)

- Procédures appelantes :

FDDTRAITEMENT
PARCOURS

- Procédures appelées :

TRANSFORMEWORD
REMPLIRWORD
TRTPAGELIST
MISERECORD
TRTPAGEDATA
PARCOURS
INCR

- Algorithme logique :

```
début
  si positionnement sur une page
    début
      inscription dans les "PARTS DIRECTORY" du début des informations
      relatives à la page courante ;
      initialisation du compteur d'informations ;
      remplissge pour la page courante de la partie "HEADER" ;
      initialisation de la longueur du "HEADER" dans l'ensemble de blocs;
      remplissage pour la page courante de la partie "DATA" ;
      initialisation de la longueur de la zone "DATA" pour l'ensemble de blocs
   fin
   iteration avec la page suivante
fin
```

- Spécification des variables :

REC	: pointeur vers un bloc du FDD, contenant l'adresse du bloc où les "PARTS DIRECTORY" doivent être mises à jour
I	: compteur
ADRDEBUTENTITYLIST	: adresse du début de l'"ENTITY LIST", (en nombre de "bytes" depuis le début du FDD
AERIERPART	: adresse de la première "PART DIRECTORY"
CPT	: compteur
LONGDATA	: nombre de mots de la "DATA LIST"
J	: compteur
VL1	: "byte", premier élément d'un "word"
VL2	: "byte", second élément d'un "word"
VL11	: "word"
IERPAS	: booléen qui indique le traitement de la première page
IER	: booléen qui indique si un nombre impair de caractères a été rencontré
ADRTABLEAU	: pointeur vers le bloc courant du FDD

3.3.1. PROCEDURE TRTPAGELIST

- Entrée:

MEMPOINTEUR : pointeur vers le record contenant la page à traiter

- Sortie :

rien

- Effet :

Cette procédure construit récursivement pour chacune des lignes d'une page, l'"ENTITY LIST" pour une page du document.
Une "ENTITY LIST" est composée d'un "HEADER" et d'une partie "DATA"
(cfr. page 6. du document "FORMATTED DOCUMENT DESCRIPTION" de R.D. KERSCH)

- Pré-condition :

(MEMPOINTEUR)

- Post-condition :

("ENTITY LIST" crée pour la page courante et insérée dans l'ensemble des blocs FDD)

- Procédure appelante :

PARCOURS

- Procédures appelées :

MISERECORD
INCR
TRANSFORMEWORD
TRTLIGNE
TRTPAGELIST

- Algorithme logique :

```
debut
   si l'on a un record de type entité
      debut
         création d'un nouvel élément dans le tableau d'adresse ;
         mémorisation de la longueur, de l'adresse ;
         création du "HEADER" de l'"ENTITY LIST"
                 (informations : type fonte,x,y,deltax,deltay) ;
         création du corps de l'"ENTITY LIST" (contenant les commandes) ;
         réinitialisation des éléments : longueur de la partie
         de commande, de l'"ENTITY LIST" dans les places précédemment
         réservées ;
      fin
   itération sur l'entité suivante
fin
```

- Spécification des variables :

MEMTAB	: pointeur vers un élememt du tableau d'adresse
FON	: entier indiquant le numéro de la fonte à indiquer dans l'"ENTITY LIST"
I,J	: compteurs
VALINTERIM	: variable utilisée pour calculer l'adresse ou indiquer la longueur du "DATA LIST"
NBCARENTITE	: nombre de caractères d'une entité
VAR1	: "byte", premier élément d'un "word"
VAR2	: "byte", deuxième élément d'un "word"
V	: "byte"
VL13	: "word"
VAL1	: pointeur vers un bloc
VAR1	: caractère à insérer dans la "DATA LIST"

3.3.2. PROCEDURE TRTPAGEDATA

- Entrees :

VAR1 : premier caractère à insérer dans la "DATA LIST" (utilité : comme on a une procédure récursive sur les bouts de lignes, il se peut que l'on ait un nombre impair de caractères)
ELEMENTCOURANT : pointeur vers le record courant de l'arbre de formatage
IER : booléen qui indique que le premier caractère d'un mot est traité sans avoir rencontré le second
IERPAS : booléen indiquant le début de la data list pour une entité
TAB : pointeur vers le dernier record d'adresse
LONGDATA : nombre de caractères déjà rencontrés dans la data list

- Sorties :

VAR1 : caractère de numéro impair rencontré
IER : booléen qui indique que le premier caractère d'un mot sans a été rencontré
IERPAS : booléen étant à true quand l'on commence la "DATA LIST" mis à jour
TAB : pointeur vers le dernier record d'adresse mis à jour
LONGDATA : nombre de caractères déjà rencontrés dans la "DATA LIST" mis à jour

- Effet :

Cette procédure récursive crée la "DATA LIST" d'une page d'un document
(cfr. page 11. du document "FORMATTED DOCUMENT DESCRIPTION" de R.D. HERSCH)

- Pré-conditions :

(
VAR1 ∧ ELEMENTCOURANT ∧ IER ∧ IERPAS ∧ PTABLEAU ∧ TAB ∧ LONGDATA
)

- Post-conditions :

(
VAR1 ∧ IER ∧ IERPAS ∧ PTABLEAU ∧ TAB ∧ LONGDATA
)

- Procédures appelantes :

TRTPAGEDATA
PARCOURS

- Procédures appelées :

TRTPAGEDTAT
TRANSFORMEWORD
REMPLIRWORD
MISERECORD

- Algorithme logique :

```
début
  si record de type bout de ligne alors
    début
        lecture de deux caractères du string associé au record ;
        ajout de ces deux caractères dans la partie "DATA" ;
    fin
  si record de type entité alors
    début
        inscription de l'adresse du "DATA LIST" de l'entité précédente
        dans l'ensemble de blocs ;
        itération sur le record successeur ;
    fin
  si record de type ligne ou bout de ligne alors
    début
        traitement du dernier caractère de la ligne (au cas où il y en
        aurait un nombre impair)
    fin
  itération sur le record suivant ;
fin
```

- Spécification des variables :

TR : variable indiquant le nombre de caractères traités
VALINTERIM : variable servant au calcul de l'adresse où insérer l'adresse de la "DATA LIST"
VAR2 : deuxième caractère à insérer dans l'ensemble de blocs
VAL1 : pointeur un élément du tableau d'adresse
VL11 : "word"
VL1 : "byte", premier d'un "word"
VL2 : "byte", second d'un "word"

3.3.3. PROCEDURE TRTLIGNE

- Entrées :

MEMLIGNE : pointeur vers le record de type ligne à traiter
NBCARENTITE : nombre de caractères de l'entité

- Sortie :

NBCARENTITE : nombre de caractères de l'entité mis à jour

- Effet :

Cette procédure crée la partie "ENTITY LIST COMMANDS"
(cfr. page 8. du document "FORMATTED DOCUMENT DESCRIPTION" de R.D. HERSCH)

- Pré-conditions :

(
(MEMLIGNE) ∧(NBCARENTITE)
)

- Post-conditions :

(
(NBCARENTITE) ∧
((un bout de ligne)∧(set_x)∧(set_y)∧(set_inc_ words pace)∧
(hightlighting)∧(fonte)
)
)

- Procédures appelantes :

TRTPAGELIST
TRTLIGNE

- Procédures appelées :

MISERECORD
TRTLIGNE

- Algorithme logique :

```
début
  tant qu'il existe des bouts de lignes associés à la ligne à traiter
    début
      réalisation du <variable_spaced_char> ;
      réalisation du <font> ;
      réalisation du <set_inc_wordspace> ;
      réalisation du <set_x> ;
      réalisation du <set_y> ;
      si le bout de ligne est souligné réalisation du <hightligthing> ;
      réalisation du <show_car> ;
      si le bout de ligne est souligné réalisation du fin <hightligthing> ;
    fin ;
  itération sur la ligne suivante ;
fin
```

- Spécification des variables :

BOUTLI	: pointeur vers un record de type bout de ligne
TYSOULI	: code indiquant le type de soulignement désiré
TYS	: variable servant au calcul de TYSOULI
TY	: variable servant au calcul de TYSOULI
FON	: code indiquant le type de fonte désiré
VAL	: variable contenant le déplacement horizontal (en nombre de fois 256)
VARINCR	: variable servant pour la détermination du <set_inc_wordspace>
VL1	: "word"
VL21	: premier "byte" du "word"
VL22	: second "byte" du "word"

3.0.1. PROCEDURE INCR

- Entrée :

INCRCOMPTEUR : variable donnant l'adresse courante dans les blocs

- Sortie :

INCR : variable contenant l'adresse courante dans les blocs augmentée de 1

- Effet :

Cette procédure incrémente le compteur d'adresse courante ; au cas où un bloc de 256 "words" est rempli (ou 512 "bytes"), un nouveau bloc est crée et chainé avec le précédent.

- Pré-condition :

(
INCRCOMPTEUR
)

- Post-conditions :

```
(
  ((COMPTEURINF <> 0)
    ∧( ((COMPTEURINF + 1) MOD 256 = 0)
      ∧(nouveau record crée)
      ∧(COMPTEURTABRECORD := COMPTEURTABRECORD + 1)
      ∧(nouveau record initialisé à moins un)
      ∧(lien du record avec le précédent effectué)
      )
    )
  ∧(INCR = INCRCOMPTEURINF + 1)
)
```

- Procédures appelantes :

MISERECORD
CONSTRDIRECTORY
TRTPAGELIST
PARCOURS

- Procédure appelée :

MISEAMOINSUN

- Algorithme logique :

```
debut
   Si un bloc est entièrement rempli alors
        debut
          création d'un nouveau bloc ;
          lien avec le précédent ;
          initialisation du bloc à moins un ;
          incrémentation du compteur de bloc ;
        fin
   incrémentation de l'adresse courante ;
fin
```

- Spécification des variables :

REC : pointeur vers le bloc précédent

3.0.2. PROCEDURE MISERECORD

- Entrées :

X : première variable à ajouter au bloc courant
Y : seconde variable à ajouter au bloc courant

- Sortie :

rien

- Effet :

Cette procédure ajoute deux entiers au bloc de mots courants. Au cas où le bloc est rempli, la procédure en crée un nouveau et fait le lien avec le précédent

- Pré-conditions :

```
(
 X ∧ Y
)
```

- Post-conditions :

```
(
  (TABRECORD*.ZONETRAVAIL[COMPTEURINF MOD 256]  contient les deux nombres
   décodés en binaire)
∧(COMPTEURINF = INCR(COMPTEURINF)
)
```

- Procédures appelantes :

CONSTRDIRECTORY
CONSTRPARTDIRECTORY
TRTPAGEDATA
TRTLIGNE
TRTPAGELIST
PARCOURS

- Procédures appelées :

INCR
REMPLIRWORD

- Algorithme logique :

```
debut
  décodage des variables en entrée afin de former un "word" ;
  ajout du "word" au bloc courant
fin
```

- Spécification des variables :

ADRESSE : variable servant au calcul de l'adresse où insérer le "word"
VL1 : "word"
VL21 : "byte", première partie du "word"
VL22 : "byte", seconde partie du "word"

3.0.3. PROCEDURE TRANSFORMEMOTS

- Entrée :

VALEURTRAVAIL : variable contenant le nombre à décoder

- Sorties :

TITI31 : premier "byte" contenant la variable decodee en binaire, "byte" de poids le plus élevé
TITI32 : second "byte" contenant la variable decodee en binaire, "byte" de poids le plus faible

- Effet :

Cette procédure transforme un entier en deux "bytes" (2 * 8 bits). Si le nombre en entrée est négatif, les deux "bytes" sont initialisés à moins un

- Pré-condition :

X

(
VALEURTRAVAIL < 65535
)

- Post-conditions :

(
(((VALEURTRAVAIL < 0)∧(TITI31[K] = 1) ∧(TITI32[K] = 1)∧(0 <= K <= 7)))
∨((VALEURTRAVAIL >= 0)∧
((((VALEURTRAVAIL - 32768 >= 0)∧ (TITI31[0] = 1))∨ (TITI31[0] = 0))
∧(((VALEURTRAVAIL - 49152 >= 0)∧ (TITI31[1] = 1)) ∨ (TITI31[1] = 0))
∧(((VALEURTRAVAIL - 57344 >= 0) ∧ (TITI31[2] = 1)) ∨ (TITI31[2] = 0))
∧(((VALEURTRAVAIL - 61440 >= 0) ∧ (TITI31[3] = 1)) ∨ (TITI31[3] = 0))
∧(((VALEURTRAVAIL - 63488 >= 0) ∧ (TITI31[4] = 1)) ∨ (TITI31[4] = 0))
∧(((VALEURTRAVAIL - 64512 >= 0) ∧ (TITI31[5] = 1)) ∨ (TITI31[5] = 0))
∧(((VALEURTRAVAIL - 65024 >= 0) ∧ (TITI31[6] = 1)) ∨ (TITI31[6] = 0))
∧(((VALEURTRAVAIL - 65280 >= 0) ∧ (TITI31[7] = 1)) ∨ (TITI31[7] = 0))
∧(((VALEURTRAVAIL - 65408 >= 0) ∧ (TITI32[0] = 1)) ∨ (TITI32[0] = 0))
∧(((VALEURTRAVAIL - 65472 >= 0) ∧ (TITI32[1] = 1)) ∨ (TITI32[1] = 0))
∧(((VALEURTRAVAIL - 65504 >= 0) ∧ (TITI32[2] = 1)) ∨ (TITI32[2] = 0))
∧(((VALEURTRAVAIL - 65520 >= 0) ∧ (TITI32[3] = 1)) ∨ (TITI32[3] = 0))
∧(((VALEURTRAVAIL - 65528 >= 0) ∧ (TITI32[4] = 1)) ∨ (TITI32[4] = 0))

∧(((VALEURTRAVAIL - 65532 >= 0) ∧ (TITI32[5] = 1)) ∨ (TITI32[5] = 0))
∧(((VALEURTRAVAIL - 65534 >= 0)∧ (TITI32[6] = 1))∨ (TITI32[6] = 0))
∧(((VALEURTRAVAIL - 65535 >= 0) ∧ (TITI32[7] = 1)) ∨ (TITI32[7] = 0))
)
)
)

- Procédures appelantes :

PARCOURS
TRTPAGELIST
TRTPAGEDATA
MISERECORD
FDDTRAITEMENT

- Procédure appelée :

rien

- Algorithme logique :

début
initialisation des deux "bytes" à '0' ;
si la variable en entrée est négative alors initialisation des deux "bytes" à moins un ;
pour toutes les valeurs clefs, test si la variable en entrée moins la valeur clef est négative, si oui initialisation du bit à un
fin

- Spécification des variables :

J, I : compteur

3.0.4. PROCEDURE REMPLIRWORD

- Entrées :

VAR1 : "byte" de poids le plus élevé
VAR2 : "byte" de poids le plus faible

- Sortie :

TITI1 : "word" contenant les deux "bytes"

- Effet :

Cette procédure rend un "word" contenant les deux "bytes" en entrée

- Pré-conditions :

(
VAR1 ∧VAR2
)

- Post-conditions :

(
(TITI[0] = VAR1)
∧(TITI[1] = VAR2)
)

- Procedures appelantes :

PARCOURS
TRTPAGELIST
MISERECORD
TRTPAGEDATA
FDDTRAITEMENT

- Procedure appelée :

rien

- Algorithme logique :

debut
initialisation des deux "bytes" du "word"
fin

- Spécification des variables :

rien

3.0.5. PROCEDURE MISEAMOINSUN

- Entrée :

MISEAMOINSUNTRAVAIL : bloc de 512 "bytes"

- Sortie :

MISEAMOINSUNTRAVAIL : bloc de 512 "bytes" mis à moins un

- Effet :

Cette procedure initialise un bloc de 512 "bytes" (256 "words") à moins un

- Pré-condition :

(
MISEAMOINSUNTRAVAIL
)

- Post-conditions :

(
((MISEAMOINSUNTRAVAIL[K] = MISEAMOINUNWORD) ∧(0 >= K >= 255))
∧((MISEAMOINSUNWORD[K] = MISEAMOINSBYTE) ∧(0 >= K >= 1))
∧((MISEAMOINSUNBYTE[K] = 1) ∧ (0 >= K >= 7))
)

- Procedures appelantes :

INCR
CONSTRDIRECTORY

- Procedure appelée :

rien

- Algorithme logique :

debut
initialisation d'un "byte" à un;
initialisation d'un "word" à la valeur du "byte" ;
initialisation du bloc à la valeur du "word"
fin

- Spécification des variables :

MISEAMOINSUNI : compteur
MISEAMOINSUNBYTE : "byte" à initialiser à la valeur -1
MISEAMOINSUNWORD : "word" à initialiser à la valeur - 1

3.0.6. PROCEDURE VALEUR

- Entrée :

VALEURTRAVAIL : variable à décoder

- Sorties :

TITI2 : "byte", resultat du décodage

- Effet :

Cette procédure décode un entier et rend sa valeur binaire

- Pré-condition :

VALEUR

- Post-conditions :

(
((VALEUR < 0)∧(TITI2[K] = 1) ∧ (0 >= K >=7))
∨
((VALEUR >= 0)
∧((VALEUR - 128 >= 0) ∧ (TITI2[0] = 1)) ∨ (TITI2[0] = 0))
∧((VALEUR - 192 >= 0) ∧ (TITI2[1] = 1)) ∨ (TITI2[1] = 0))
∧((VALEUR - 224 >= 0) ∧ (TITI2[2] = 1)) ∨ (TITI2[2] = 0))
∧((VALEUR - 240 >= 0) ∧ (TITI2[3] = 1)) ∨ (TITI2[3] = 0))
∧((VALEUR - 248 >= 0) ∧ (TITI2[4] = 1)) ∨ (TITI2[4] = 0))
∧((VALEUR - 252 >= 0) ∧ (TITI2[5] = 1)) ∨ (TITI2[5] = 0))
∧((VALEUR - 254 >= 0) ∧ (TITI2[6] = 1)) ∨ (TITI2[6] = 0))
∧((VALEUR - 255 >= 0) ∧ (TITI2[7] = 1)) ∨ (TITI2[7] = 0))
)
)

- Procédure appelante :

MISERECORD

- Procédure appelée :

rien

- Algorithme logique :

```
debut
  initialisation du "byte" à 'o' ;
  si la valeur de la variable en entrée est négative initialisation
  du "byte" à '1' ;
  pour toutes les valeurs clefs
   si la valeur de la variable en entrée moins la valeur clef est négative,
   initialisation d'un bit à moins un
fin
```

- Spécification des variables :

I : compteur

4. PROCEDURE IMPRBIT

- Entree :

  PTPRRE : pointeur vers les blocs FDD

- Sortie :

  rien

- Effet :

  Cette procédure construit le fichier binaire qui comprendra toutes les indications de formatage en format FDD, à partir de l'ensemble de blocs

  contenant ces informations et se trouvant en mémoire centrale.

- Pré-condition :

  (
  PTABRECORD
)

- Post-condition :

  (
  fichier DEFINI contenant toutes les indications de formatage
)

- Procédure appelante :

  PROGRAMME PRINCIPAL

- Procédure appelée :

  TRANSFORME

- Algorithme logique :

```
début
  ouverture du fichier DEFIMI en écriture ;
  tant que le pointeur vers un bloc n'est pas à nil
    début
      lecture d'un bloc;
      inscription du bloc dans le fichier ;
      initialisation du pointeur vers le bloc suivant ;
    fin
fin
```

- Spécification des variables :

```
COMPTEUR     : compteur du nombre de word d'un bloc
IMP1         : indicateur de word
IMP2         : indicateur de byte
IMPVARIABLE  : indicateur de bloc
IMPINTERIM   : variable de travail
IMPZEROUN    : variable de valeur o ou 1
IMPBYTE      : byte lu dans un bloc de word
IMPWORD      : word lu dans un bloc
```

4.1. PROCEDURE TRANSFORME

- Entrée :

  V2 : nombre : un ou zero

- Sortie :

  V1 : caractère un ou zéro

- Effet :

  Cette procédure transforme un nombre en caractère

- Pré-condition :

  (
  V2
)

- Post-condition :

  (
  V1
)

- Procédure appelante :

  IMPRBIT

- Procédure appelée :

  rien

- Algorithme logique :

```
début
  si le nombre est '1' alors le caractere vaut un, sinon le
       caractère vaut zero
fin
```

- Spécification des variables :

  rien

0.1. PROCEDURE ERREUR

- Entrée :

NUMERO : numéro de l'erreur

- Sortie :

rien

- Effet :

Cette procédure écrit dans un fichier d'erreur une phrase en fonction du numéro spécifié. Le numéro apparait également sur l'écran.

- Pré-condition :

(
NUMERO
)

- Post-conditions :

Message apparaissant dans un fichier source en fonction du numéro de l'erreur.

Liste des erreurs avec leur numéro

SIGLES : GLO : erreur dans une séquence d'une commande globale
LOC : erreur dans une séquance d'une commande locale

NUMERO	TEXTE
1	rencontre de la fin du fichier source
2	niveau de chapitre > 4
3	erreur dans la création d'un ordre
7	introduction de caractères <> de "\\","\d","\D" en entete de fichier
8	introduction de caractère(s) au niveau du chapitre avant l'entête
9	decalage d'un paragraphe spécifié dans un élement d'un autre type
15	rencontre de trois bons alors que l'on n'est pas dans une liste
16	séquence lexicale non reconnue
17	séquence d'implémentation non reconnue
20	caractère spécial non reconnu
30	GLO redefinition de la fonte pour les titres
31	GLO redefinition de la fonte pour la date
32	GLO redefinition de la fonte pour le résumé
33	GLO redefinition de la fonte pour l'auteur
34	GLO redefinition de la fonte pour les entêtes de niveau 1
35	GLO redefinition de la fonte pour les entêtes de niveau 2
36	GLO redefinition de la fonte pour les entêtes de niveau 3
37	GLO redefinition de la fonte pour les entêtes de niveau 4
38	GLO redefinition de la fonte pour les textes de figure
39	GLO redefinition de la fonte pour les listes
40	GLO redefinition de la fonte pour les paragraphes
42	GLO redefinition de la fonte pour les notes bas de pages
43	GLO redefinition de la fonte pour la table des matieres
44	GLO redefinition de la fonte - élément non reconnu
45	GLO redefinition de la police pour les titres
46	GLO redefinition de la police pour la date
47	GLO redefinition de la police pour le résumé
48	GLO redefinition de la police pour l'auteur
49	GLO redefinition de la police pour les entêtes de niveau 1
50	GLO redefinition de la police pour les entêtes de niveau 2
51	GLO redefinition de la police pour les entêtes de niveau 3
52	GLO redefinition de la police pour les entêtes de niveau 4
53	GLO redefinition de la police pour les textes de figure
54	GLO redefinition de la police pour les listes
55	GLO redefinition de la police pour les paragraphes
56	GLO redefinition de la police pour les notes bas de pages
57	GLO redefinition de la police pour la table des matieres
58	GLO redefinition de la police - élément non reconnu
59	GLO redefinition du fait qu'un élément doit être centré
60	GLO redefinition du fait qu'un élément doit être justifié
61	GLO redefinition de l'interligne pour les titres
62	GLO redefinition de l'interligne pour la date
63	GLO redefinition de l'interligne pour le résumé
64	GLO redefinition de l'interligne pour l'auteur
65	GLO redefinition de l'interligne pour les entêtes de niveau 1
66	GLO redefinition de l'interligne pour les entêtes de niveau 2
67	GLO redefinition de l'interligne pour les entêtes de niveau 3
68	GLO redefinition de l'interligne pour les entêtes de niveau 4
69	GLO redefinition de l'interligne pour les textes de figure
70	GLO redefinition de l'interligne pour les listes
71	GLO redefinition de l'interligne pour les paragraphes
72	GLO redefinition de l'interligne pour les notes bas de pages
73	GLO redefinition de l'interligne pour la table des matieres
74	GLO redefinition de l'interligne - élément non reconnu
75	GLO redefinition du soulignement pour les titres
76	GLO redefinition du soulignement pour la date
77	GLO redefinition du soulignement pour le résumé
78	GLO redefinition du soulignement pour l'auteur
79	GLO redefinition du soulignement pour les entêtes de niveau 1
80	GLO redefinition du soulignement pour les entêtes de niveau 2
81	GLO redefinition du soulignement pour les entêtes de niveau 3
82	GLO redefinition du soulignement pour les entêtes de niveau 4
83	GLO redefinition du soulignement pour les textes de figure
84	GLO redefinition du soulignement pour les listes
85	GLO redefinition du soulignement pour les paragraphes
86	GLO redefinition du soulignement pour les notes bas de pages
87	GLO redefinition du soulignement pour la table des matieres
88	GLO redefinition du soulignement - élément non reconnu
90	GLO redefinition de la marge gauche pour les titres
91	GLO redefinition de la marge gauche pour la date
92	GLO redefinition de la marge gauche pour le résumé
93	GLO redefinition de la marge gauche pour l'auteur
94	GLO redefinition de la marge gauche pour les entêtes de niveau 1
95	GLO redefinition de la marge gauche pour les entêtes de niveau 2
96	GLO redefinition de la marge gauche pour les entêtes de niveau 3
97	GLO redefinition de la marge gauche pour les entêtes de niveau 4
98	GLO redefinition de la marge gauche pour les textes de figure
99	GLO redefinition de la marge gauche pour les listes
100	GLO redefinition de la marge gauche pour les paragraphes
101	GLO redefinition de la marge gauche pour les notes bas de pages
102	GLO redefinition de la marge droite pour les titres
103	GLO redefinition de la marge droite pour la date
104	GLO redefinition de la marge droite pour le résumé
105	GLO redefinition de la marge droite pour l'auteur
106	GLO redefinition de la marge droite pour les entêtes de niveau 1
107	GLO redefinition de la marge droite pour les entêtes de niveau 2
108	GLO redefinition de la marge droite pour les entêtes de niveau 3
109	GLO redefinition de la marge droite pour les entêtes de niveau 4
110	GLO redefinition de la marge droite pour les textes de figure
111	GLO redefinition de la marge droite pour les listes

```
112     GLO redéfinition de la marge droite pour les paragraphes
113     GLO redéfinition de la marge droite pour les notes bas de pages
181     GLO redéfinition de la marge droite pour la table des matières
182     GLO redéfinition de la marge supérieure
183     GLO redéfinition de la marge inférieure
114     GLO marge gauche/marge droite des titres trop grande
115     GLO marge gauche/marge droite de la date trop grande
116     GLO marge gauche/marge droite du resumé trop grande
117     GLO marge gauche/marge droite de l'auteur  trop grande
118     GLO marge gauche/marge droite des entêtes de niveau 1 trop grande
119     GLO marge gauche/marge droite des entêtes de niveau 2 trop grande
120     GLO marge gauche/marge droite des entêtes de niveau 3 trop grande
121     GLO marge gauche/marge droite des entêtes de niveau 4 trop grande
122     GLO marge gauche/marge droite des textes de figure trop grande
123     GLO marge gauche/marge droite des listes trop grande
124     GLO marge gauche/marge droite des paragraphes trop grande
125     GLO marge gauche/marge droite des note bas de pages trop grande
126     GLO marge gauche/marge droite de la table des matières trop grande
127     GLO marge supérieure/marge inférieure trop grande
128     GLO redéfinition de la marge gauche - élément non reconnu
129     GLO redéfinition de la marge droite - élément non reconnu
130     GLO redéfinition de la marge gauche pour la table des matières
131     GLO redéfinition de la marge droite pour la table des matières
132     GLO initialisation du numéro des entetes de niveau 1
133     GLO initialisation du numéro des pages
134     GLO redéfinition du decalage de la premiere ligne d'un paragraphe
135     GLO redéfinition du décalage d'une liste
136     GLO redéfinition du nombre de millimètres entres éléments
137     GLO redéfinition du décalage - spécification du type d'élement incorrect
138     GLO initialisation du numéro des figures
139     GLO initialisation du numero des notes bas de page
140     GLO initialisation - spécification du type d'élément incorrect
141     GLO redéfinition du type d'implémentation des listes
142     GLO redéfinition du nombre de niveau à indiquer dans une table des matieres
143     GLO redéfinition non reconnue
145     GLO impression réduite
180     GLO redéfinition de la marge gauche de la table des matières
200     Rencontré d'un chapitre de niveau 2 incorrect
201     Rencontre d'un chapitre de niveau 3 incorrect
202     Rencontre d'un chapitre de niveau 4 incorrect
203     LOC rencontre de '\\C_' où '_' n'est pas défini correctement
205     LOC erreur dans la spécification de la police
206     LOC rencontre de '\\P_' où '_' n'est pas défini correctement
207     LOC erreur dans la spécification de la fonte
208     LOC erreur dans le code de fin, soit de soulignement, soit de police,
        soit de fonte
209     LOC rencontre de '\\F_' où '_' n'est pas défini correctement
210     LOC erreur dans la spécification de l'implémentation d'une liste
211     LOC rencontre de '\\L_' où '_' n'est pas défini correctement
212     LOC erreur dans la spécification du type de soulignement
213     LOC erreur dans la spécification du décalage d'un paragraphe
214     LOC erreur dans la spécification de la marge gauche
215     LOC erreur dans la spécification de la marge droite
216     LOC type de marge non reconnue
217     LOC erreur dans une spécification d'implémentation
220     LOC erreur dans la spcification d'un decalage de liste
300     LOC "mot" de plus de 24 caracteres
```

- Procédures appelantes :

  Un grand nombre de procédures appellent ce module

- Procédure appelée :

  rien

- Algorithme logique

```
debut
  inscription d'une erreur dans le fichier d'erreur en fonction du numéro
fin
```

- Spécification des variables :

  rien

0.2. PROCEDURE PARCOURSARBSTRUC

- Entrée :

  COLONEL : pointeur vers la racine de l'arbre de structure

- Sortie :

  rien

- Effet :

  Cette procedure permet d'imprimer dans le fichier erreurs le contenu de l'arbre de structure

- Pre-condition :

  (
  COLONEL
)

- Post-condition :

  (
  tous les elements de tous les records de l'arbre de structure se trouvent dans le fichier erreurs
)

- Procedure appelante :

  PROGRAMME PRINCIPAL

- Procedure appelee :

  rien

- Algorithme logique :

```
debut
  pour tous les records de l'arbre de structure
    debut
      impression de la valeur des variables
    fin
fin
```

- Spécification des variables :

  rien

0.3. PROCEDURE PARCOURSARBPOR

- Entrée :

  POINTEUR : pointeur vers la racine de l'arbre de formatage

- Sortie :

  rien

- Effet :

  Cette procedure, à partir de l'adresse de la racine de l'arbre de stucture, parcourt récursivement celui-ci et inscrit dans le fichier résultat RESPOR certains éléments de l'arbre afin d'effectuer des tests de comparaisons entre le résultat effectif et le résultat attendu

- Pré-condition :

  (
  (POINTEUR <> NIL)
)

- Post-conditions :

  (
  insertion dans le fichier RESPOR des éléments désirés
)

- Procédure appelante :

  PROGRAMME PRINCIPAL

- Procédure appelée :

  rien

- Algorithme logique

```
debut
  si le pointeur vers l'entité <> nil
    debut
      insertion des éléments résultats dans le fichier RESPOR ;
      parcours sur le pointeur successeur ;
      parcours sur le pointeur suivant
    fin
fin
```

- Spécification des variables :

  rien

ANNEXE B

PROCEDURES PASCAL

INTRODUCTION

Cette annexe se compose de :

- le plan de l'architecture physique du système. Le plan de l'architecture physique du système se compose de :
  - la liste des noms de procédures
  - le numéro correspondant de l'architecture physique (annexe a)
  - le numéro de la page où se trouve la spécification de la procédure (annexe a)
  - le numéro de la page où se trouve la procédure
- les procédures PASCAL

Remarque : Les procédures PASCAL ne comprennent aucun commentaire, il faut se référer à l'annexe a.

1. Plan de l'architecture physique

	numero de la procedure	page de l'annexe A	page de l'annexe B
PROGRAM PRINCIPAL		A.10	B.6
Procedure PARCOURSABSTRUC	0.2	A.95	B.8
Procedure PARCOURSARBFORM	0.3	A.96	B.8
Procedure ERREUR	0.1	A.93	B.9
Procedure ARBRESTRUCTURE	1.	A.14	B.12
Procedure LECTURECARACTERE	1.0.1.	A.36	B.12
Procedure LECPOINT	1.0.5.	A.38	B.12
Procedure LECSLASH	1.0.4.	A.37	B.13
Procedure LECDEUXPOINT	1.0.2.	A.36	B.13
Procedure LECJUSTBLANC	1.0.3.	A.37	B.13
Procedure DECODAGEVALEUR	1.0.6.	A.38	B.13
Procedure EXPDIX	1.0.7.	A.39	B.13
Procedure DEFELEMBASE	1.1.	A.16	B.13
Procedure PROMODBOOL	1.2.1.	A.18	B.14
Procedure PROMODREDUIT	1.2.2.	A.17	B.14
Procedure PROMODAUTRE	1.2.3.	A.19	B.15
Procedure PROMARGE	1.2.4.	A.21	B.18
Procedure PRODIVERS	1.2.5.	A.22	B.19
Procedure MODELEMBASE	1.2.	A.17	B.21
Procedure PARCOURSFINNOTE	1.3.6.	A.29	B.21
Procedure FINNOTEBASPAGE	1.3.7	A.29	B.22
Procedure RECHERCHERNIVEAU	1.3.4.	A.28	B.22
Procedure RECHERCHERDIMENTION	1.3.5.	A.28	B.22
Procedure IMPLEMLISTE	1.4.4.	A.35	B.23
Procedure NUMEROTATIONENTETE	1.4.3.	A.34	B.23
Procedure NUMEROTATIONFIG	1.4.1.	A.32	B.24
Procedure NUMEROTATIONNOTE	1.4.2.	A.33	B.24
Procedure LIEN	1.3.2.	A.26	B.25
Procedure CREERRECORD	1.3.1.	A.25	B.26

2. PROCEDURES PASCAL

2.1. Introduction

Ce point reprend l'ensemble des textes PASCAL des procédures. Comme il a été signalé dans l'introduction, la plupart des procédures ne contiennent aucun commentaire. Seules certaines procédures plus complexes contiennent quelques commentaires afin de faciliter leur compréhension.

Les textes PASCAL des procédures sont dans leur ordre logique de création.

2.2. Textes PASCAL des Procédures

```
(* LES COMMENTAIRES ET LES SPECIFICATIONS DES PROCEDURES SE TROUVENT DANS
  L'ANNEXE A DU MEMOIRE CONSACRE A LA REALISATION DE CE FORMATEUR *)


program PRINCIPAL (r10,r11,i11,g11,r14,
                   input,output,resfor,erreurs,source,res,format,defini);

label   1;
const   nbpointmm                   = 9.448;
        rognage                     = 5;
type    typefonte                   = (italique,gras,normal);
        typesoulignement            = (rien,continu,discontinu,superieure);
        typeliste                   = (etoile,point,tiret);
        typefeuille                 = (titre,auteur,abstract,date,document,liste,
                                       chapitre,paragraphe,figure,textefigure,
                                       entete,notebaspage,up,down);
        tyfor                       = (pagetypo,entitetypo,lignetypo,
                                       boutlignetypo);
        genrefeuille                = (a4,b5);
        pmot                        = ^mot;
        pgeneral                    = ^general;
        ptypographique              = ^typographique;
        tex                         = file of char;
        ensemble                    = packed array[1..128] of char;
        tab                         = array[1..4] of pgeneral;

        general = record
                  centrage                  : boolean;
                  margegauche               : integer;
                  margedroite               : integer;
                  margesuperieure           : integer;
                  margeinferieure           : integer;
                  largeur                   : integer;
                  hauteur                   : integer;
                  longueurfeuille           : integer;
                  largeurfeuille            : integer;
                  polan                     : 1..20;
                  police                    : 1..20;
                  niveauchapitre            : integer;
                  decalageparagraphe        : integer;
                  premiere                  : integer;
                  derniere                  : integer;
                  fonan                     : typefonte;
                  fonte                     : typefonte;
                  interentident             : integer;
                  interentdocu              : integer;
                  interligne                : integer;
                  index                     : boolean;
                  impreduite                : boolean;
                  numerotation              : boolean;
                  superieure                : boolean;
                  seule                     : boolean;
                  colonnage                 : boolean;
                  pagination                : boolean;
                  souan                     : typesoulignement;
                  soulignement              : typesoulignement;
                  tfeuille                  : genrefeuille;
                  liste                     : typeliste;
                  tyfeuille                 : typefeuille;
                  sautpage                  : boolean;
                  sautligne                 : boolean;
                  tablematiere              : boolean;
                  mrdmat                    : integer;
                  mrgmat                    : integer;
                  poltab                    : integer;
                  fotab                     : typefonte;
                  intertab                  : integer;
                  soutab                    : typesoulignement;
                  centab                    : boolean;
                  nivtab                    : integer;
                  numpage                   : integer;
                  pointsuccess1             : pgeneral;
                  pointsuccess2             : pgeneral;
                  pointsuivant              : pgeneral;
                  pointmot                  : pmot;
                  end;

        mot = record
                  soulignement              : typesoulignement;
                  police                    : 1..20;
                  string                    : packed array[0..24]of char;
                  fonte                     : typefonte;
                  pointmot                  : pmot;
                  sautligne                 : boolean;
                  end;

        typographique = record
                  typesorte                 : typefeuille;
                  typo                      : tyfor;
                  x                         : integer;
                  y                         : integer;
                  deltax                    : integer;
                  deltay                    : integer;
                  policetypographique       : 1..20;
                  fontetypographique        : typefonte;
                  soulignementtypographique : typesoulignement;
                  stringtypographique       : packed array [0..128] of char;
```

```
                nombreblanc             : integer;
                largeurvoulue           : integer;
                nombreelement           : integer;
                valuespace              : integer;
                interligneentite        : integer;
                sautpagetypographique   : boolean;
                pointsuccess            : ptypographique;
                pointsuivant            : ptypographique;
                end;

        ptabrecord              = ^tabrecor;
        zeroun                  = Ø..1;
        byte                    = packed array[Ø..7] of zeroun;
        word                    = packed array[Ø..1] of byte;
        block                   = packed array[Ø..255] of word;
        tabrecor        = record
                zonetravail             : block;
                pointsuivant            : ptabrecord;
                numero                  : integer
                end;


var adressepremierrecord : pgeneral;
    format,defini,r1Ø,r11,l11,g11,r14,resfor,erreurs,source,res : text;
    pagetypographique,adressepremiertypographique,mpadressepremierepage :
                ptypographique;
    adresse : ptabrecord;

    (* indicateur de saut de page *)

    csautaut,csautdat,csauten1,csauten2,csauten3,csauten4,csautfig,
    csautlis,csautnot,csautpar,csautres,csauttit,csauttab,

    (* indicateur de centrage des elements de structure *)

    ccentaut,ccentdat,ccenten1,ccenten2,ccenten3,ccenten4,
    ccentfig,ccentlis,ccentnot,ccentpar,ccentres,ccenttit,ccenttab,

    (* indicateur booleen divers *)

    test,ccolodoc,cimpredu,cindexdo,cpagidoc,cpagisup,
    cseulpag,ctabdoc : boolean;

    (* valeurs entieres diverses *)

    cdecpara,cdecalis,cinferie,cdernier,cinteren,clargeur,
    clongueur,cnumfig,cnumnot,cnumpag,cnument,ctabniv,

    (* valeur entiere de l'interligne des elements de structure *)

    cinliaut,cinlidat,cinlien1,cinlien2,cinlien3,cinlien4,
    cinlifig,cinlilis,cinlinot,cinlipar,cinlires,cinlitit,cinlitab,

    (* valeur entiere de la marge gauche des elements de structure *)

    cmargaut,cmargdat,cmargen1,cmargen2,cmargen3,cmargen4,cmargfig,
    cmarglis,cmargnot,cmargpar,cmargres,cmargtit,cmargtab,

    (* valeur entiere de la marge droite des elements de structure *)

    cmardaut,cmarddat,cmarden1,cmarden2,cmarden3,cmarden4,
    cmardfig,cmardlis,cmardnot,cmardpar,cmardres,cmardtit,cmardtab,

    (* valeur entiere de la marge superieure / inferieure *)

    cmardsud,cmardind : integer;
    proportionblanc : real;

    (* type de fonte pour les elements de structure *)

    cfonaute,cfondate,cfonenn1,cfonenn2,cfonenn3,cfonenn4,cfonlist,cfonnote,
    cfonfigu,cfonpara,cfonresu,cfontitr,cfontabl    : typefonte ;


    (* type d'implementation pour la liste *)

    cimplist : typeliste;

    (* type de police pour les elements de structure *)

    cpolaut,cpoldat,cpolen1,cpolen2,cpolen3,cpolen4,cpollis,cpolnot,
    cpolfig,cpolpar,cpolres,cpoltit,cpoltab    : 1..2Ø ;

    (* type de soulignement pour les elements de structure *)

    csouaut,csoudat,csouen1,csouen2,csouen3,csouen4,csoulis,csounot,
    csoufig,csoupar,csoures,csoutit,csoutab    : typesoulignement ;

    (* type de feuille *)

    ctypefeui : genrefeuille ;
```

```
(*****************************************************************************)
(*                        DECLARATIONS DES PROCEDURES                        *)
(*****************************************************************************)

procedure PARCOURSARBSTRUC (colonel : pgeneral);

        procedure IMPRIME( pointeurversmot :pmot);

        var i : integer;

        begin
          while pointeurversmot <> nil do
           begin
             i:= 1;
             while i <= ord(pointeurversmot^.string[0]) do
                begin
                  write(erreurs,pointeurversmot^.string[i]);
                  i := i + 1;
                end;
              writeln(erreurs,'##');
              write(erreurs, 'TYPE DE SOULIGNEMENT DU MOT : ');
              writeln(erreurs,pointeurversmot^.soulignement);
              write(erreurs,'TYPE DE FONTE DU MOT : ');
              writeln(erreurs,pointeurversmot^.fonte);
              write(erreurs,'TYPE DE POLICE DU MOT : ');
              writeln(erreurs,pointeurversmot^.police);
              write(erreurs,'INDICATEUR DE SAUT DE LIGNE : ');
              if pointeurversmot^.sautligne = true then
                writeln(erreurs,'vrai') else writeln(erreurs,'false');
              pointeurversmot := pointeurversmot^.pointmot;
           end;
        end;

begin
  if colonel <> nil then
    begin
      writeln(erreurs,'==================================================');
      write(erreurs,colonel^.tyfeuille);
      write(erreurs,'  ');
      if colonel^.centrage then writeln(erreurs,'cet element est centre')
         else writeln(erreurs,'cet element n est PAS CENTRE');
      if colonel^.tyfeuille = chapitre then writeln(erreurs,
              colonel^.niveauchapitre) else writeln(erreurs,'  ');
      if colonel^.tyfeuille = liste then
        begin
          write(erreurs, ' TYPE LISTE :');
          writeln(erreurs,colonel^.liste);
        end;
      if colonel^.tyfeuille = paragraphe then
        begin
          write(erreurs,'GRANDEUR DU DECALAGE : ') ;
          writeln(erreurs,colonel^.decalageparagraphe);
        end;
      write(erreurs,' INDICATEUR DE SAUT DE PAGE  : ');
      if colonel^.sautpage = true then
        writeln(erreurs,'vrai') else writeln(erreurs,'false');
      write(erreurs,' MARGE GAUCHE : ');
      writeln(erreurs,colonel^.margegauche);
      write(erreurs,' MARGE DROITE : ');
      writeln(erreurs,colonel^.margedroite);
      if colonel^.pointmot <> nil then IMPRIME(colonel^.pointmot);
      PARCOURSARBSTRUC(colonel^.pointsuccess1);
      PARCOURSARBSTRUC(colonel^.pointsuccess2);
      PARCOURSARBSTRUC(colonel^.pointsuivant)
    end
end;
(*****************************************************************************)
procedure PARCOURSARBFORM(pointeur :ptypographique);

var i : integer;
begin
 if pointeur <> nil then
   begin
     write(resfor,' TYPE : ');write(resfor,pointeur^.typo);writeln(resfor);
     write(resfor,' X :');write(resfor,pointeur^.x);
     write(resfor,' Y :');write(resfor,pointeur^.y);
     write(resfor,' DELTAX :');write(resfor,pointeur^.deltax);
     write(resfor,' DELTAY :');write(resfor,pointeur^.deltay);
     write(resfor,'LARGEURLIGNE : ');write(resfor,pointeur^.largeurvoulue);
     write(resfor,'NOMBRELEMENT : ');write(resfor,pointeur^.nombreelement);
     writeln(resfor);
     write(resfor,' VALEUR BLANC : ');write(resfor,pointeur^.valuespace);
     write(resfor,' NOMBRE BLANC : ');write(resfor,pointeur^.nombreblanc);
     writeln(resfor);
     i := 1;
     while i <= pointeur^.nombreelement do
        begin
          write(resfor,pointeur^.stringtypographique[i]);
          i := i + 1;
        end;
     writeln(resfor);
     PARCOURSARBFORM(pointeur^.pointsuccess);
     PARCOURSARBFORM(pointeur^.pointsuivant);
   end;
end;
(*****************************************************************************)
```

```
procedure ERREUR(numero : integer);

begin
  writeln(numero);
  case numero of

1  : writeln(erreurs,'RENCONTRE DE LA FIN DU FICHIER SOURCE');
7  : writeln(erreurs,'INSERTTION DE CARACTERES INCORRECTS EN DEBUT DE FICHIER');


30 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LE TITRE');
31 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LA DATE');
32 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LE RESUME');
33 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR L''AUTEUR');
34 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 1');

35 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 2');

36 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 3');

37 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES ENTETES DE NIVEAU 4');

38 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES TEXTES DE FIGURE');


39 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES LISTES');
40 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES PARAGRAPHES');
42 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LES NOTES BAS DE PAGE');
43 : writeln(erreurs,'REDEFINITION DE LA FONTE POUR LA TABLE DES MATIERES');
44 : writeln(erreurs,'REDEFINITION DE LA FONTE - ELEMENT INCORRECT');
45 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LE TITRE');
46 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LA DATE');
47 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LE RESUME');
48 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR L''AUTEUR');
49 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 1');

50 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 2');

51 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 3');

52 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES ENTETES DE NIVEAU 4');

53 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES TEXTES DE FIGURE');

54 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES LISTES');
55 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES PARAGRAPHES');
56 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LES NOTES BAS DE PAGE');
57 : writeln(erreurs,'REDEFINITION DE LA POLICE POUR LA TABLE DES MATIERES');
58 : writeln(erreurs,'REDEFINITION DE LA POLICE - ELEMENT NON RECONNU');
59 : writeln(erreurs,'REDEFINITION DU FAIT QU''UN ELEMENT DOIT ETRE CENTRE');
60 : writeln(erreurs,'REDEFINITION DU FAIT QU''UN ELEMENT DOIT ETRE JUSTIFIE');

61 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LE TITRE');
62 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LA DATE');
63 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LE RESUME');
64 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR L''AUTEUR');
65 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES ENTETES DE NIVEAU 1');
67 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES ENTETES DE NIVEAU 3');
68 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES ENTETES DE NIVEAU 4');
69 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES TEXTES DE FIGURE');


70 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES LISTES');
71 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES PARAGRAPHES');
72 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LES NOTES BAS DE PAGE');
73 : writeln(erreurs,'REDEFINITION INTERLIGNE POUR LA TABLE DES MATIERES');
74 : writeln(erreurs,'REDEFINITION INTERLIGNE - ELEMENT NON RECONNU');
75 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LE TITRE');
76 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LA DATE');
77 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LE RESUME');
78 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR L''AUTEUR');
79 : writeln(erreurs,'REDEFINITION SOULIGNEMENT POUR LES ENTETES DE NIVEAU 1');

80 : writeln(erreurs,'REDEFINITION SOULIGNEMENT POUR LES ENTETES DE NIVEAU 2');

81 : writeln(erreurs,'REDEFINITION SOULIGNEMENT POUR LES ENTETES DE NIVEAU 3');

82 : writeln(erreurs,'REDEFINITION SOULIGNEMENT POUR LES ENTETES DE NIVEAU 4');

83 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LES TEXTES DE FIGURE');

84 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LES LISTES');
85 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LES PARAGRAPHES');
86 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LES NOTES BAS DE PAGE');


87 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT POUR LA TABLE DES MATIERES');


88 : writeln(erreurs,'REDEFINITION DU SOULIGNEMENT - ELEMENT NON RECONNU');
91 : writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE POUR LA DATE');
92 : writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE POUR LE RESUME');
93 : writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE POUR L''AUTEUR');
94 : writeln(erreurs,'REDEFINITION MARGE GAUCHE POUR LES ENTETES DE NIVEAU 1');

95 : writeln(erreurs,'REDEFINITION MARGE GAUCHE POUR LES ENTETES DE NIVEAU 2');

96 : writeln(erreurs,'REDEFINITION MARGE GAUCHE POUR LES ENTETES DE NIVEAU 3');
```

```
97 : writeln(erreurs,'REDEFINITION MARGE GAUCHE POUR LES ENTETES DE NIVEAU 4');

98 : writeln(erreurs,'REDEFINITION MARGE GAUCHE POUR LES TEXTES DE FIGURE');
99 : writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE POUR LES LISTES');
100: writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE POUR LES PARAGRAPHES');
101: writeln(erreurs,'REDEFINITION MARGE GAUCHE POUR LES NOTES BAS DE PAGE');
102: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR LE TITRE');
103: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR LA DATE');
104: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR LE RESUME');
105: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR L''AUTEUR');
106: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LES ENTETES DE NIVEAU 1');

107: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LES ENTETES DE NIVEAU 2');

108: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LES ENTETES DE NIVEAU 3');

109: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LES ENTETES DE NIVEAU 4');

110: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LES TEXTES DE FIGURE');
111: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR LES LISTES');
112: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR LES PARAGRAPHES');
113: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LES NOTES BAS DE PAGE');
181: writeln(erreurs,'REDEFINITION MARGE DROITE POUR LA TABLE DES MATIERES');
182: writeln(erreurs,'REDEFINITION DE LA MARGE SUPERIEURE');
183: writeln(erreurs,'REDEFINITION DE LA MARGE INFERIEURE');
114: writeln(erreurs,'MARGE GAUCHE/MARGE DROITE DU TITRE TROP GRANDE');
115: writeln(erreurs,'MARGE GAUCHE/MARGE DROITE DE LA DATE TROP GRANDE');
116: writeln(erreurs,'MARGE GAUCHE/MARGE DROITE DU RESUME TROP GRANDE');
117: writeln(erreurs,'MARGE GAUCHE/MARGE DROITE DE L''AUTEUR TROP GRANDE');
118: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 1');
       writeln(erreurs,' TROP GRANDE')
     end;
119: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 2');
       writeln(erreurs,' TROP GRANDE')
     end;
120: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 3');
       writeln(erreurs,' TROP GRANDE')
     end;
121: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DES ENTETES DE NIVEAU 4');
       writeln(erreurs,' TROP GRANDE')
     end;
122: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DES TEXTES DE FIGURES');
       writeln(erreurs,' TROP GRANDE')
     end;
123: writeln(erreurs,'MARGE GAUCHE/MARGE DROITE DES LISTES TROP GRANDE');
124: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DES NOTES BAS DE PAGE');
       writeln(erreurs,' TROP GRANDE')
     end;
125: writeln(erreurs,'MARGE GAUCHE/MARGE DROITE DES PARAGRAPHES TROP GRANDE');
 126: begin
       write(erreurs,'MARGE GAUCHE/MARGE DROITE DE LA TABLE DES MATIERES');
       writeln(erreurs,' TROP GRANDE')
     end;
127: begin
       write(erreurs,'MARGE SUPERIEURE/MARGE INFERIEURE DE LA TABLE DES');
       writeln(erreurs,' MATIERES TROP GRANDE')
     end;
128: writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE - ELEMENT NON RECONNU');
 129: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE - ELEMENT NON RECONNU');
 130: writeln(erreurs,'REDEFINITION DE LA MARGE GAUCHE POUR LA TABLE DES MATIERES

');
131: writeln(erreurs,'REDEFINITION DE LA MARGE DROITE POUR LA TABLE DES MATIERES

');
132: writeln(erreurs,'INITIALISATION DU NUMERO DES PAGES');
133: writeln(erreurs,'INITIALISATION DU NUMERO DE L''ENTETE DE NIVEAU 1');
134: begin
       write(erreurs,'REDEFINITION DU DECALAGE DE LA PREMIERE LIGNE DES');
       writeln(erreurs,' PARAGRAPHES')
     end;
135: writeln(erreurs,'REDEFINITION DU DECALAGE DES LISTES');
136: writeln(erreurs,'REDEFINITION DU NOMBRE DE MILLIMETRES ENTRE ELEMENTS');
137: begin
       write(erreurs,'REDEFINITION DU DECALAGE - SPECIFICATION DU TYPE');
       writeln(erreurs,' D''ELEMENT INCONNU')
     end;
138: writeln(erreurs,'INITIALISATION DU NUMERO DES FIGURES');
139: writeln(erreurs,'INITIALISATION DU NUMERO DES NOTES BAS DE PAGE');
140: begin
       write(erreurs,'INITIALISATION DU NUMERO - SPECIFICATION DU TYPE');
       writeln(erreurs,' D''ELEMENT INCONNU')
     end;
141: writeln(erreurs,'REDEFINITION DU TYPE D''IMPLEMENTATION DES LISTES');
142: writeln(erreurs,'DEFINITION DU NOMBRE DE NIVEAU DE LA TABLE DES MATIERES');


143: writeln(erreurs,'REDEFINITION NON RECONNUE');
145: writeln(erreurs,'IMPRESSION REDUITE');
2  : writeln(erreurs,'NIVEAU DE CHAPITRE > 4');
3  : writeln(erreurs,'ERREUR DANS LA CREATION D''UN ORDRE');
8  : begin
       write(erreurs,'INTRODUCTION DE CARACTERE(S) AU NIVEAU DU CHAPITRE');
       writeln(erreurs,' AVANT L''ENTETE')
     end;
9  : begin
       write(erreurs,'DECALAGE DU PARAGRAPHE OU D''UNE LISTE SPECIFIE');
       writeln(erreurs,' DANS UN ELEMENT D''UN AUTRE TYPE')
     end;
15 : writeln(erreurs,'RENCONTRE DE TROIS \ ALORS QU''ON N''EST PAS DANS UNE LIST
```

```
16 : writeln(erreurs,'SEQUENCE LEXICALE NON RECONNUE');
17 : writeln(erreurs,'SEQUENCE D''IMPLEMENTATION NON RECONNUE');
2Ø : writeln(erreurs,'CARACTERE SPECIAL NON RECONNU');
2ØØ: writeln(erreurs,'RENCONTRE D''UN CHAPITRE DE NIVEAU 2 INCORRECT');
2Ø1: writeln(erreurs,'RENCONTRE D''UN CHAPITRE DE NIVEAU 3 INCORRECT');
2Ø2: writeln(erreurs,'RENCONTRE D''UN CHAPITRE DE NIVEAU 4 INCORRECT');
2Ø3: begin
       write(erreurs,'RENCONTRE DE: "\\C_" OU: "_" N''EST PAS DEFINI');
       writeln(erreurs,' CORRECTEMENT')
     end;
2Ø4: writeln(erreurs,'ERREUR DANS LA SPECIFICATION DE LA POLICE');
2Ø6: begin
       write(erreurs,'RENCONTRE DE: "\\P_" OU: "_" N''EST PAS DEFINI');
       writeln(erreurs,' CORRECTEMENT')
     end;
2Ø7: writeln(erreurs,'ERREUR DANS LA SPECIFICATION DE LA FONTE');
2Ø8: begin
       write(erreurs,'ERREUR DANS LE CODE DE FIN DE SOULIGNEMENT SOIT DE');
       writeln(erreurs,' POLICE SOIT DE FONTE')
     end;
2Ø9: begin
       write(erreurs,'RENCONTRE DE: "\\F_" OU: "_" N''EST PAS DEFINI');
       writeln(erreurs,' CORRECTEMENT')
     end;
21Ø: writeln(erreurs,'ERREUR DANS LA SPECIFICATION D''IMPLEMENTATION D''UNE LIST

E');
211: begin
       write(erreurs,'RENCONTRE DE: "\\L_" OU: "_" N''EST PAS DEFINI');
       writeln(erreurs,' CORRECTEMENT')
     end;
212: writeln(erreurs,'ERREUR DANS LA SPECIFICATION DU TYPE DE SOULIGNEMENT');
213: writeln(erreurs,'ERREUR DANS LA SPECIFICATION DU DECALAGE D''UN PARAGRAPHE'

);
214: writeln(erreurs,'ERREUR DANS LA SPECIFICATION DE LA MARGE GAUCHE');
  215: begin
       write(erreurs,'ERREUR DANS LA SPECIFICATION DE LA MARGE');
       writeln(erreurs,' DROITE')
     end;
216: writeln(erreurs,'TYPE DE MARGE NON RECONNU');

217: writeln(erreurs,'ERREUR DANS UNE COMMANDE D''IMPLEMENTATION');
22Ø: writeln(erreurs,'ERREUR DANS LA SPECIFICATION DU DECALAGE D''UNE LISTE');
  end;
end;
(*****************************************************************************)
```

```
(*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)
(*:::::::::::::::::::ARBRE DE STRUCTURE:::::::::::::::::::::::::::::::::::::::::*)
(*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)

procedure  ARBRESTRUCTURE(var adrierrecord : pgeneral);

type quatreentiers = array[1..4] of integer;

var
    impl,fanion,fin,creer,trouver,iermot,ierpas,dejasep,creerliste,newliste,
    notdejaslash,memnewliste,carspec,
    sautauteur,sautdate,sautentete1,sautentete2,sautentete3,sautentete4,
    sautfigure,sautliste,sautnote,sautparagraphe,sautresume,sauttitre,
    sauttable,
    centreauteur,centredate,centreentniv1,centreentniv2,centreentniv3,
    centreentniv4,centrefigure,centreliste,centrenote,centrepara,centreresume,
      centretitre,centretable,
    colonnagedocu,impreduite,indexdocu,paginationdocu,paginationsuperieure,
    seulesurpage,tablematieredocu : boolean;
    decapara,decaliste,inferieure,derniere,interentite,largeurfeuilledocu,
    longueurfeuilledocu,numeroflg,numeronote,numeropage,numeroentete,
    tableniv,i,ind1,ind2,
    inliauteur,inlidate,inlientniv1,inlientniv2,inlientniv3,inlientniv4,
    inlifigure,inlliste,inlinote,inlipara,inliresume,inlititre,inlitable,
    margegaucheauteur,margegauchedate,margegaucheentniv1,margegaucheentniv2,
    margegaucheentniv3,margegaucheentniv4,margegauchefigure,
    margegauchelliste,margegauchenote,margegauchepara,margegaucheresume,
    margegauchetitre,margegauchetable,
    margedroiteauteur,margedroitedate,margedroiteentniv1,margedroiteentniv2,
    margedroiteentniv3,margedroiteentniv4,margedroitefigure,
    margedroitelliste,margedroitenote,margedroitepara,margedroiteresume,
    margedroitetitre,margedroitetable,
    margesupdocu,margeinfdocu : integer;
    fonteauteur,fontedate,fonteentniv1,fonteentniv2,fonteentniv3,
    fonteentniv4,fonteliste,fontenote,fontefigure,fontepara,fonteresume,
    fontetitre,fontetable   : typefonte ;
    impliste : typeliste;
    policeauteur,policedate,policeentniv1,policeentniv2,policeentniv3,
    policeentniv4,policeliste,policenote,policefigure,policepara,
    policeresume,policetitre,policetable   : 1..20 ;
    soulauteur,soulidate,soulientniv1,soulientniv2,soulientniv3,
    soulientniv4,soulliste,soulinote,soulifigure,soulipara,souliresume,
    soulititre,soultable   : typesoulignement ;
    typefeuilledocu : genrefeuille ;
    tableau : quatreentiers;
    ierrecord,prec,precp,precpp : pgeneral;
    memmot :pmot;
    ch :char;

(*****************************************************************************)
(*                DECLARATIONS DES PROCEDURES                                *)
(*****************************************************************************)
(*****************************************************************************)
(*                          UTILITAIRES                                      *)
(*****************************************************************************)

procedure LECTURECARACTERE(var ch :char; var fin :boolean);

begin
  fin := false;
  if not eof(source) then
    begin
      if not eoln(source) then read(source,ch)
        else
        begin
          readln(source);
          read(source,ch)
        end;
    end;
  if eof(source) then  begin fin := true;goto 1 end;
end;

(*****************************************************************************)

procedure LECPOINT;

var ch : char;
    fin : boolean;

begin
  LECTURECARACTERE(ch,fin);
  while ch <> '.' do  LECTURECARACTERE(ch,fin);
end;

(*****************************************************************************)
```

```
procedure LECSLASH;

var ch :char;
    fin : boolean;

begin
  LECTURECARACTERE(ch,fin);
  while ch <> '\' do LECTURECARACTERE(ch,fin);
end;
(*****************************************************************************)

procedure LECDEUXPOINT;

var ch :char;
    fin : boolean;

begin
  LECTURECARACTERE(ch,fin);
  while ch <> ':' do  LECTURECARACTERE(ch,fin)
end;
(*****************************************************************************)

procedure LECJUSBLANC;

var ch :char;
    fin : boolean;

begin
  LECTURECARACTERE(ch,fin);
  while ch <> ' ' do  LECTURECARACTERE(ch,fin)
end;
(*****************************************************************************)

procedure DECODAGEVALEUR(var nb : integer;var ch :char);

var fin : boolean;

begin
  nb := -1;
  while ((ord(ch) >= 48 ) and (ord(ch) <= 57)) do
    begin
      if nb = -1 then nb := ord(ch) - 48 else nb := (nb*10)+(ord(ch) -48);
      LECTURECARACTERE(ch,fin);
    end;
end;
(*****************************************************************************)

FUNCTION EXPDIX(valeur : integer): integer;

var val ,i : integer;

begin
  val := 1;
  for i := 1 to valeur do val := val * 10;
  EXPDIX := val;
end;

(*****************************************************************************)
(*              DEFINTION DES VARIABLES                                      *)
(*****************************************************************************)

procedure DEFELEMBASE;

begin
sautauteur           := csautaut;      sautdate             := csautdat;
sautentete1          := csauten1;      sautentete2          := csauten2;
sautentete3          := csauten3;      sautentete4          := csauten4;
sautfigure           := csautfig;      sautliste            := csautlis;
sautnote             := csautnot;      sautparagraphe       := csautpar;
sautresume           := csautres;      sauttitre            := csauttit;
sauttable            := csauttab;      centreauteur         := ccentaut;
centredate           := ccentdat;      centreentniv1        := ccenten1;
centreentniv2        := ccenten2;      centreentniv3        := ccenten3;
centreentniv4        := ccenten4;      centrefigure         := ccentfig;
centrelliste         := ccentlis;      centrenote           := ccentnot;
centrepara           := ccentpar;      centreresume         := ccentres;
centretitre          := ccenttit;      centretable          := ccenttab;
colonnagedocu        := ccolodoc;      impreduite           := cimpredu;
indexdocu            := cindexdo;      paginationdocu       := cpagidoc;
paginationsuperieure:= cpagisup;       seulesurpage         := cseulpag;
tablematieredocu     := ctabdoc;       decapara             := cdecpara;
decaliste            := cdecalis;      inferieure           := cinferie;
derniere             := cdernier;      interentite          := cinteren;
largeurfeuilledocu   := clargeur;      longueurfeuilledocu:= clongueur;
numerofig            := cnumfig;       numeronote           := cnumnot;
numeropage           := cnumpag;       numeroentete         := cnument;
tableniv             := ctabniv;       inlilauteur          := cinliaut;
inlildate            := cinlidat;      inlientniv1          := cinlien1;
inlientniv2          := cinlien2;      inlientniv3          := cinlien3;
inlientniv4          := cinlien4;      inlifigure           := cinlifig;
inlilliste           := cinlilis;      inlinote             := cinlinot;
inlipara             := cinlipar;      inliresume           := cinlires;
inlititre            := cinlitit;      inlitable            := cinlitab;
```

```
margegaucheauteur  := cmargaut;        margegauchedate    := cmargdat;
margegaucheentniv1 := cmargen1;        margegaucheentniv2 := cmargen2;
margegaucheentniv3 := cmargen3;        margegaucheentniv4 := cmargen4;
margegauchefigure  := cmargfig;        margegauchelliste  := cmarglis;
margegauchenote    := cmargnot;        margegauchepara    := cmargpar;
margegaucheresume  := cmargres;        margegauchetitre   := cmargtit;
margegauchetable   := cmargtab;        margedroiteauteur  := cmardaut;
margedroitedate    := cmarddat;        margedroiteentniv1 := cmarden1;
margedroiteentniv2 := cmarden2;        margedroiteentniv3 := cmarden3;
margedroiteentniv4 := cmarden4;        margedroitefigure  := cmardfig;
margedroitelliste  := cmardlis;        margedroitenote    := cmardnot;
margedroitepara    := cmardpar;        margedroiteresume  := cmardres;
margedroitetitre   := cmardtit;        margedroitetable   := cmardtab;
margesupdocu       := cmardsud;        margeinfdocu       := cmardind;
fonteauteur        := cfonaute;        fontedate          := cfondate;
fonteentniv1       := cfonenn1;        fonteentniv2       := cfonenn2;
fonteentniv3       := cfonenn3;        fonteentniv4       := cfonenn4;
fontelliste        := cfonlist;        fontenote          := cfonnote;
fontefigure        := cfonfigu;        fontepara          := cfonpara;
fonteresume        := cfonresu;        fontetitre         := cfontitr;
fontetable         := cfontabl;        implliste          := cimplist;
policeauteur       := cpolaut;         policedate         := cpoldat;
policeentniv1      := cpolen1;         policeentniv2      := cpolen2;
policeentniv3      := cpolen3;         policeentniv4      := cpolen4;
policelliste       := cpollis;         policenote         := cpolnot;
policefigure       := cpolfig;         policepara         := cpolpar;
policeresume       := cpolres;         policetitre        := cpoltit;
policetable        := cpoltab;         soulliauteur       := csouaut;
soulidate          := csoudat;         soulientniv1       := csouen1;
soulientniv2       := csouen2;         soulientniv3       := csouen3;
soulientniv4       := csouen4;         soullliste         := csoulis;
soulinote          := csounot;         soulifigure        := csoufig;
soulipara          := csoupar;         soulresume         := csoures;
soulititre         := csoutit;         soulitable         := csoutab;
                    typefeuilledocu    := ctypefeui;
end;

{*************************************************************************}

(*      RECONNAISSANCE POUR LA MODIFICATION DES varIABLES GLOBALES       *)

{*************************************************************************}

procedure PROMODBOOL(c1,c2 : char);

begin
  case c1 of
    'c' : colonnagedocu := true;
    'i' : indexdocu := true;
    'p' : begin
            paginationdocu := true;
            ch := 't';
            while ((ch <> '.') and (ch <> ':')) do LECTURECARACTERE(ch,fin);
            if ch= ':' then
              begin
                while ch = ' ' do  LECTURECARACTERE(ch,fin);
                case ch of
                  's','S' : paginationsuperieure := true;
                end;
              end;
          end;
    't' : begin
            tablematieredocu := true;
            paginationdocu := true;
          end;
    'u' : seulesurpage := true;
  end;
  LECSLASH
end;

{*************************************************************************}

procedure PROMODREDUIT;

label 3;

var ch :char;
    fin : boolean;
    nb : integer;

begin
  LECDEUXPOINT;
  LECTURECARACTERE(ch,fin);
  while ch = ' ' do  LECTURECARACTERE(ch,fin);
  DECODAGEVALEUR(nb,ch);
  inferieure := nb;
  if nb < 0 then goto 3;
  while ch <> ',' do LECTURECARACTERE(ch,fin);
  LECTURECARACTERE(ch,fin);
  while ch = ' ' do  LECTURECARACTERE(ch,fin);
  DECODAGEVALEUR(nb,ch);
  derniere := nb;
  if inferieure > derniere then ERREUR(145) else impreduite := true;
  if ch <> '\' then LECSLASH;
3 : if inferieure < 0 then ERREUR(145);
end;

{*************************************************************************}
```

```
procedure PROMODAUTRE(c1,c2 : char);

label 2,3,4,5,6;

var ch,niv,ch1,ch2: char;
    nb : integer;
    fin,bool,pas,test : boolean;
    tysou : typesoulignement;
    tyfon : typefonte;

begin
  pas := false;
  nb := -1;
  LECDEUXPOINT;
  LECTURECARACTERE(ch,fin);
  while ch = ' ' do  LECTURECARACTERE(ch,fin);
  ch1 := ch;
       (* ch1 est le premier caractere de ce qui suit les premiers ':' *)
  LECTURECARACTERE(ch2,fin);
       (* ch2 est le deuxieme caractere de ce qui suit les premiers ':' , son
       utilite est de determiner si l'on a un element logique de type
       titre ou de type table de matiere *)

  case c1 of

    'c','j' : begin
                if ((c1= 'j' ) or (ch='J')) then bool := false
                                            else bool := true;
              case  ch1 of
                'a','A' : centreauteur := bool;
                'd','D' : centredate := bool;
                'e','E' : begin
                            LECDEUXPOINT;
                            LECDEUXPOINT;
                            ch := ' ';
                            while ch = ' ' do LECTURECARACTERE(ch,fin);
                            case ch of
                              '1' : centreentniv1 := bool;
                              '2' : centreentniv2 := bool;
                              '3' : centreentniv3 := bool;
                              '4' : centreentniv4 := bool;
                             otherwise pas := true;
                            end;
                          end;
                'f','F' : centrefigure := bool;
                'l','L' : centreliste := bool;
                'n','N' : centrenote := bool;
                'p','P' : centrepara := bool;
                'r','R' : centreresume := bool;
                't','T' : begin
                            if ch2 = 'A' then ch2 := 'a';
                            if ch2 = 'I' then ch2 := 'i';
                            case ch2 of
                              'a' : centretable := bool;
                              'i' : centretitre := bool;
                             otherwise pas := true;
                            end;
                          end;
                  otherwise pas := true;
              end;
              if ((pas) and (c1 = 'j')) then ERREUR(60);
              if ((pas) and (c1 = 'c')) then ERREUR(59);
        end;

    'f' : begin
            if ((ch1 = 'e') or (ch1 = 'E')) then
              begin
                LECDEUXPOINT;
                LECDEUXPOINT;
                LECTURECARACTERE(ch,fin);
                while ch = ' ' do  LECTURECARACTERE(ch,fin);
                niv := ch;
                if ((ord(niv) < 48) or (ord(niv) > 51)) then goto 2;
              end;
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do LECTURECARACTERE(ch,fin);
            case ch of
              'n','N' : tyfon := normal;
              'i','I' : tyfon := italique;
              'g','G' : tyfon := gras;
             otherwise pas := true;
            end;
            case  ch1 of
              'a','A' : if pas then ERREUR(33) else fonteauteur := tyfon;
              'd','D' : if pas then ERREUR(31) else fontedate := tyfon;
              'e','E' : case niv of
                          '1' : if pas then ERREUR(34)
                                       else fonteentniv1 := tyfon;
                          '2' : if pas then ERREUR(35)
                                       else fonteentniv2 := tyfon;
                          '3' : if pas then ERREUR(36)
                                       else fonteentniv3 := tyfon;
                          '4' : if pas then ERREUR(37)
                                       else fonteentniv4 := tyfon;
                         otherwise ERREUR(44);
                        end;
```

```
            'f','F' : if pas then ERREUR(38) else fontefigure := tyfon;
            'l','L' : if pas then ERREUR(39) else fonteliste := tyfon;
            'n','N' : if pas then ERREUR(41) else fontenote := tyfon;
            'p','P' : if pas then ERREUR(40) else fontepara := tyfon;
            'r','R' : if pas then ERREUR(32) else fonteresume := tyfon;
            't','T' : case ch2 of
                        'a','A' : if pas then ERREUR(42) else
                                    fontetable := tyfon;
                        'i','I' : if pas then ERREUR(30) else
                                    fontetitre := tyfon;
                      otherwise ERREUR(44);
                      end;
          otherwise ERREUR(44);
          end;
        end;

  'p' : begin
          if ((ch1 = 'e') or (ch1 = 'E')) then
            begin
              LECDEUXPOINT;
              LECDEUXPOINT;
              LECTURECARACTERE(ch,fin);
              while ch = ' ' do LECTURECARACTERE(ch,fin);
              niv := ch;
              if ((ord(niv) < 48) or (ord(niv) >= 52)) then goto 3;
            end;
          LECDEUXPOINT;
          LECTURECARACTERE(ch,fin);
          while ch = ' ' do LECTURECARACTERE(ch,fin);
          DECODAGEVALEUR(nb,ch);
          case nb of
            10 : pas := false;
            11 : pas := false;
            14 : pas := false;
            otherwise pas := true;
          end;
          if pas then nb := 11;
          case ch1 of
            'a','A' : if pas then ERREUR(48) else policeauteur := nb;
            'd','D' : if pas then ERREUR(46) else policedate := nb;
            'e','E' : case niv of
                        '1' : if pas then ERREUR(49)
                                     else policeentniv1 := nb;
                        '2' : if pas then ERREUR(50)
                                     else policeentniv2 := nb;
                        '3' : if pas then ERREUR(51)
                                     else policeentniv3 := nb;
                        '4' : if pas then ERREUR(52)
                                     else policeentniv4 := nb;
                      otherwise ERREUR(58);
                      end;
            'f','F' : if pas then ERREUR(53) else policefigure := nb;
            'l','L' : if pas then ERREUR(54) else policeliste := nb;
            'n','N' : if pas then ERREUR(56) else policenote := nb;
            'p','P' : if pas then ERREUR(55) else policepara := nb;
            'r','R' : if pas then ERREUR(47) else policeresume := nb;
            't','T' : case ch2 of
                        'a','A' : if pas then ERREUR(58) else
                                    policetable := nb;
                        'i','I' : if pas then ERREUR(45) else
                                    policetitre := nb;
                      otherwise ERREUR(58);
                      end;
          otherwise ERREUR(58);
          end;
        end;

  's' : begin
          if ((ch1 = 'e') or (ch1 = 'E')) then
            begin
              LECDEUXPOINT;
              LECDEUXPOINT;
              LECTURECARACTERE(ch,fin);
              while ch = ' ' do   LECTURECARACTERE(ch,fin);
              niv := ch;
              if ((ord(niv) < 48) or (ord(niv) >= 52)) then goto 4;
            end;
          LECDEUXPOINT;
          LECTURECARACTERE(ch,fin);
          while ch = ' ' do LECTURECARACTERE(ch,fin);
          case ch of
            'c','C' : tysou := continu;
            'd','D' : tysou := discontinu;
            's','S' : tysou := superieure;
            'r','R' : tysou := rien;
            otherwise pas := true;
          end;
          case ch1 of
            'a','A' : if pas then ERREUR(78) else soulauteur := tysou;
            'd','D' : if pas then ERREUR(76) else souldate := tysou;
            'e','E' : case niv of
                        '1' : if pas then ERREUR(79)
                                     else soulentniv1 := tysou;
                        '2' : if pas then ERREUR(80)
                                     else soulentniv2 := tysou;
                        '3' : if pas then ERREUR(81)
                                     else soulentniv3 := tysou;
                        '4' : if pas then ERREUR(82)
                                     else soulentniv4 := tysou;
                      otherwise ERREUR(88);
```

```
                    end;
                'f','F' : if pas then ERREUR(83) else soulifigure := tysou;
                'l','L' : if pas then ERREUR(84) else soullliste := tysou;
                'n','N' : if pas then ERREUR(86) else soulinote := tysou;
                'p','P' : if pas then ERREUR(85) else soulipara := tysou;
                'r','R' : if pas then ERREUR(77) else soulirésume := tysou;
                't','T' : case ch2 of
                            'a','A' : if pas then ERREUR(87) else
                                                 soulitable := tysou;
                            'i','I' : if pas then ERREUR(75) else
                                                 soulititre := tysou;
                              otherwise ERREUR(88);
                          end;
              otherwise ERREUR(88);
            end;
        end;

    'i' : begin
            if ((ch1 = 'e') or (ch1 = 'E')) then
              begin
                LECDEUXPOINT;
                LECDEUXPOINT;
                LECTURECARACTERE(ch,fin);
                while ch = ' ' do LECTURECARACTERE(ch,fin);
                niv := ch;
               if ((ord(niv) < 48) or (ord(niv) >= 52)) then goto 5;
              end;
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do LECTURECARACTERE(ch,fin);
            DECODAGEVALEUR(nb,ch);
            if ((nb<0) or (nb>100)) then pas := true;
            case  ch1 of
              'a','A' : if pas then ERREUR(64) else inliauteur := nb;
              'd','D' : if pas then ERREUR(62) else inlidate := nb;
              'e','E' : case niv of
                          '1' : if pas then ERREUR(65)
                                       else inlientniv1 := nb;
                          '2' : if pas then ERREUR(66)
                                       else inlientniv2 := nb;
                          '3' : if pas then ERREUR(67)
                                       else inlientniv3 := nb;
                          '4' : if pas then ERREUR(68)
                                       else inlientniv4 := nb;
                          otherwise ERREUR(74);
                        end;
              'f','F' : if pas then ERREUR(69) else inlifigure := nb;
              'l','L' : if pas then ERREUR(70) else inlilliste := nb;
              'n','N' : if pas then ERREUR(72) else inlinote := nb;
              'p','P' : if pas then ERREUR(71) else inlipara := nb;
              'r','R' : if pas then ERREUR(63) else inliresume := nb;
              't','T' : case ch2 of
                          'a','A' : if pas then ERREUR(73) else
                                                 inlitable := nb;
                          'i','I' : if pas then ERREUR(61) else
                                                 inlititre := nb;

                            otherwise ERREUR(74);
                        end;
            otherwise ERREUR(74);
          end;
      end;
  goto 6;
2 : if ((ord(niv) < 48) or (ord(niv) > 51)) then
      begin ERREUR(44); goto 6; end;
3 : if ((ord(niv) < 48) or (ord(niv) > 51)) then
      begin ERREUR(58); goto 6; end;
4 : if ((ord(niv) < 48) or (ord(niv) > 51)) then
      begin ERREUR(88); goto 6; end;
5 : if ((ord(niv) < 48) or (ord(niv) > 51)) then
      begin ERREUR(74); goto 6; end;
6 : if ch <> '\' then LECSLASH;
end;

(*************************************************************************************)
```

```
procedure PROMARGE;

label 2,3;
var ch,typem,ch1,ch2 : char;
    valeur : integer;
    fin : boolean;

begin
  valeur := -1;
  LECDEUXPOINT;
  LECTURECARACTERE(ch,fin);
  while ch = ' ' do   LECTURECARACTERE(ch,fin);
  case ch of
    'g','G' : typem := 'g';
    'd','D' : typem := 'd';
    'i','I' : typem := 'w';
    's','S' : typem := 's';
    otherwise goto 3;
  end;
  LECDEUXPOINT;
  LECTURECARACTERE(ch,fin);
  while ch = ' ' do  LECTURECARACTERE(ch,fin);
  if ((typem = 'g') or (typem = 'd')) then
    begin
      ch1 := ch;
      LECTURECARACTERE(ch2,fin);
      if fin then ERREUR(1);
      if ((ch1 = 'e') or (ch1 = 'E')) then
        begin
          LECDEUXPOINT;
          LECDEUXPOINT;
          LECTURECARACTERE(ch,fin);
          while ch = ' ' do LECTURECARACTERE(ch,fin);
          ch2 := ch;
          if ((ord(ch2) < 48) or (ord(ch2) > 51)) then goto 2;
        end;
      LECDEUXPOINT;
      LECTURECARACTERE(ch,fin);
      while ch = ' ' do  LECTURECARACTERE(ch,fin);
    end;
  DECODAGEVALEUR(valeur,ch);
3 :
   case typem of
    's' : if valeur > Ø then margesupdocu := valeur else ERREUR(132);
    'w' : if valeur > Ø then margeinfdocu := valeur else ERREUR(183);
    'd' : case  ch1 of
            'a','A' : if valeur < Ø then ERREUR(1Ø5)
                                    else margedroiteauteur := valeur;
            'd','D' : if valeur < Ø then ERREUR(1Ø3)
                                    else margedroitedate := valeur;
            'e','E' : case ch2 of
                        '1' : if valeur < Ø then ERREUR(1Ø6)
                                            else margedroiteentniv1 := valeur;
                        '2' : if valeur < Ø then ERREUR(1Ø7)
                                            else margedroiteentniv2 := valeur;
                        '3' : if valeur < Ø then ERREUR(1Ø8)

                                            else margedroiteentniv3 := valeur;
                        '4' : if valeur < Ø then ERREUR(1Ø9)
                                            else margedroiteentniv4 := valeur;
                        otherwise ERREUR(129);
                      end;
            'f','F' : if valeur < Ø then ERREUR(11Ø)
                                    else margedroitefigure := valeur;
            'l','L' : if valeur < Ø then ERREUR(111)
                                    else margedroiteliste := valeur;
            'n','N' : if valeur < Ø then ERREUR(113)
                                    else margedroitenote := valeur;
            'p','P' : if valeur < Ø then ERREUR(112)
                                    else margedroitepara := valeur;
            'r','R' : if valeur < Ø then ERREUR(1Ø4)
                                    else margedroiteresume := valeur;
            't','T' : case ch2 of
                        'a','A' : if valeur < Ø then ERREUR(131) else
                                                margedroitetable := valeur;
                        'i','I' : if valeur < Ø then ERREUR(1Ø2) else
                                                margedroitetitre := valeur;
                        otherwise ERREUR(129);
                      end;
```

```
            otherwise ERREUR(129);
          end;
    'g' : case ch1 of
            'a','A' : if valeur < Ø then ERREUR(93)
                      else margegaucheauteur := valeur;
            'd','D' : if valeur < Ø then ERREUR(91)
                      else margegauchedate := valeur;
            'e','E' : case ch2 of
                        '1' : if valeur < Ø then ERREUR(94)
                              else margegaucheentniv1 := valeur;
                        '2' : if valeur < Ø then ERREUR(95)
                              else margegaucheentniv2 := valeur;
                        '3' : if valeur < Ø then ERREUR(96)
                              else margegaucheentniv3 := valeur;
                        '4' : if valeur < Ø then ERREUR(97)
                              else margegaucheentniv4 := valeur;
                        otherwise ERREUR(128);
                      end;
            'f','F' : if valeur < Ø then ERREUR(98)
                      else margegauchefigure := valeur;
            'l','L' : if valeur < Ø then ERREUR(99)
                      else margegaucheliste := valeur;
            'n','N' : if valeur < Ø then ERREUR(1Ø1)
                      else margegauchenote := valeur;
            'p','P' : if valeur < Ø then ERREUR(1ØØ)
                      else margegauchepara := valeur;
            'r','R' : if valeur < Ø then ERREUR(92)
                      else margegaucheresume := valeur;
            't','T' : case ch2 of
                        'a','A' : if valeur < Ø then ERREUR(13Ø) else
                                  margegauchetable := valeur;
                        'i','I' : if valeur < Ø then ERREUR(9Ø) else
                                  margegauchetitre := valeur;
                        otherwise ERREUR(128);
                      end;
            otherwise ERREUR(128);
          end;
    otherwise ERREUR(216);
  end;
  if (largeurfeuilledocu - margedroitetitre - margegauchetitre < Ø)
     then ERREUR(114);
  if (largeurfeuilledocu - margedroitedate - margegauchedate < Ø)
     then ERREUR(115);
  if (largeurfeuilledocu - margedroiteresume - margegaucheresume < Ø)
     then ERREUR(116);
  if (largeurfeuilledocu - margedroiteauteur - margegaucheauteur < Ø)
     then ERREUR(117);
  if (largeurfeuilledocu - margedroiteentniv1 - margegaucheentniv1 < Ø)
     then ERREUR(118);
  if (largeurfeuilledocu - margedroiteentniv2 - margegaucheentniv2 < Ø)

     then ERREUR(119);
  if (largeurfeuilledocu - margedroiteentniv3 - margegaucheentniv3 < Ø)
     then ERREUR(12Ø);
  if (largeurfeuilledocu - margedroiteentniv4 - margegaucheentniv4 < Ø)
     then ERREUR(121);
  if (largeurfeuilledocu - margedroitefigure - margegauchefigure < Ø)
     then ERREUR(122);
  if (largeurfeuilledocu - margedroiteliste - margegaucheliste < Ø)
     then ERREUR(123);
  if (largeurfeuilledocu - margedroitepara - margegauchepara < Ø)
     then ERREUR(124);
  if (largeurfeuilledocu - margedroitenote - margegauchenote < Ø)
     then ERREUR(125);
  if (largeurfeuilledocu - margedroitetable - margegauchetable < Ø)
     then ERREUR(126);
  if (longueurfeuilledocu - margeinfdocu - margesupdocu < Ø)
     then ERREUR(127);
2 : if ((ord(ch2) < 49) or (ord(ch2) > 52)) then ERREUR(216);
  if ch <> '\' then LECSLASH
end;

{*****************************************************************************}

procedure PRODIVERS(c1,c2 : char);

var ch,typev,typee : char;
    valeur : integer;
    bool,fin : boolean;

begin
  valeur := -1;
  bool := false;
  if c1 = 't' then
    begin
      while ((ch <> '.' ) and (ch <> ':')) do LECTURECARACTERE(ch,fin);
      if ch = '.' then bool := true;
    end
      else
    begin
      LECDEUXPOINT;
    end;
```

```
  if not bool then
    begin
      LECTURECARACTERE(ch,fin);
      while ch = ' ' do LECTURECARACTERE(ch,fin);
    end;
  case cl of
    'd' : begin
            typee := ch;
            if ch = 'P' then typee := 'p';
            if ch = 'L' then typee := 'l';
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do  LECTURECARACTERE(ch,fin);
            DECODAGEVALEUR(valeur,ch);
            case typee of
              'P', 'p' : if valeur > Ø then decapara := valeur else ERREUR(134);

              'L', 'l' : if valeur > Ø then decaliste := valeur
                                       else ERREUR(135);
              otherwise ERREUR(136);
            end;
          end;
    'i' : begin
            typee := ch;
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do   LECTURECARACTERE(ch,fin);
            typev := ch;
            case typee of
              'l','L' : case typev of

                          'e' : implliste := etoile;
                          't' : implliste := tiret;
                          'p' : implliste := point;
                          otherwise ERREUR(141);
                        end;
              otherwise ERREUR(141);
            end;
          end;
    'n' : begin
            typee := ch;
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do LECTURECARACTERE(ch,fin);
            if ((typee = 'e') or (typee = 'E')) then
              while ((ord(ch) < 48) or (ord(ch) > 57)) do
                LECTURECARACTERE(ch,fin);
            DECODAGEVALEUR(valeur,ch);
            case typee of
              'f','F' : if valeur > Ø then numerofig := valeur
                                      else ERREUR(138);
              'N','n' : if valeur > Ø then numeronote := valeur
                                      else ERREUR(139);
              'P','p' : if valeur > Ø then numeropage := valeur - 1
                                      else ERREUR(132);
              'E','e' : if valeur > Ø then numeroentete := valeur - 1
                                      else ERREUR(133)
              otherwise ERREUR(14Ø);
            end;
          end;
    't' : begin
            paginationdocu := true;
            tablematieredocu := true;
            if not bool then
              begin
                DECODAGEVALEUR(valeur,ch);
                if valeur > Ø then tableniv := valeur else ERREUR(142);
                if tableniv > 4  then
                  begin
                    tableniv := 4 ;
                    ERREUR(142);
                  end;
              end;
          end;
    'e' : begin
            DECODAGEVALEUR(valeur,ch);
            if valeur > Ø then interentite := valeur else ERREUR(137);
          end;
  end;
  if ch <> '\' then LECSLASH
end;

'*************************************************************************************)
```

```
procedure MODELEMBASE;

var ch :char;
    tampon : packed array [1..2] of char;
    fin : boolean;

begin
  LECTURECARACTERE(ch,fin);
  while ch = ' ' do LECTURECARACTERE(ch,fin);
  tampon[1] := ch;
  LECTURECARACTERE(ch,fin);
  tampon[2] := ch;
  if ((tampon <> '\\') and (tampon <> '\d') and (tampon <>'\D'))then ERREUR(7)
     else
    begin
    while ((tampon <> '\d') and  (tampon <> '\D')) do
      begin
        LECTURECARACTERE(ch,fin);
        tampon[1] := ch;
        LECTURECARACTERE(ch,fin);
        tampon[2] := ch;
        case tampon[1] of
          'c','C' : case tampon[2] of
                      'e','E' : PROMODAUTRE('c','e');
                      'o','O' : PROMODBOOL('c','o');
                     otherwise ERREUR(143);
                    end;
          'd','D' : PRODIVERS('d','e');
          'e','E' : PRODIVERS('e','l');
          'f','F' : PROMODAUTRE('f','o');
          'i','I' : case tampon[2] of
                      'm','M' : begin
                                  LECTURECARACTERE(ch,fin);
                                  LECTURECARACTERE(ch,fin);
                                  case ch of
                                    'L','l' : PRODIVERS('i','m');
                                    'R','r' : PROMODREDUIT;
                                    otherwise ERREUR(143);
                                  end;
                                end;
                      'n','N' : begin
                                  LECTURECARACTERE(ch,fin);
                                  case ch of
                                    't','T' : PROMODAUTRE('i','n');
                                    'd','D' : PROMODBOOL('i','n');
                                    otherwise ERREUR(143);
                                  end;
                                end;
                     otherwise ERREUR(143);
                    end;
          'j','J' : PROMODAUTRE('j','u');
          'm','M' : PROMARGE;
          'n','N' : PRODIVERS('n','u');
          'p','P' : case tampon[2] of
                      'o','O' : PROMODAUTRE('p','o');
                      'a','A' : PROMODBOOL('p','a');
                     otherwise ERREUR(143);
                    end;
          's','S' : PROMODAUTRE('s','o');
          't','T' : case tampon[2] of
                      'a','A' : PRODIVERS('t','a');
                      'y','Y' : ;
                     otherwise ERREUR(143);
                    end;
          'u','U' : PROMODBOOL('u','n');
          otherwise ERREUR(143);
        end;
        tampon[1] := '\';
        LECTURECARACTERE(ch,fin);
        tampon[2] := ch;
      end;
    end;
  LECPOINT;
end;

(*****************************************************************************)
(*                        PROCEDURES DIVERSES                                *)
(*****************************************************************************)

procedure PARCOURSFINNOTE(var memrec : pgeneral;prec : pgeneral);

begin
if prec <> nil then
  begin
    if prec^.tyfeuille <> notebaspage then memrec := prec;
    parcoursfinnote(memrec,prec^.pointsuccess1);
    parcoursfinnote(memrec,prec^.pointsuccess2);
```

```
    parcoursfinnote(memrec,prec^.pointsuivant);
  end;
end;
(*-------------------------------------------------------------------------------*)

procedure finNOTEBASPAGE(var prec :pgeneral);

var adressemot : pmot;

begin
  parcoursfinnote(prec,adrferrecord);
  LECPOINT;
  adressemot := prec^.pointmot;
  while adressemot <> nil do
    begin
      memmot := adressemot;
      adressemot := adressemot^.pointmot
    end;
end;
(*-------------------------------------------------------------------------------*)

procedure RECHERCHERNIVEAU (var ind1 : integer);

var ch : char;
    fin,trouv : boolean;

begin
  trouv := false;
  while not trouv do
    begin
      LECTURECARACTERE(ch,fin);
      if ((ord(ch) >= 48) and (ord(ch) <= 57 )) then
        begin
          ind1 := ord(ch) - 48;
          trouv := true;
        end;
    end;
  if ch <> '.' then LECPOINT;
  if ind1 > 4 then ERREUR(2);
end;
(*-------------------------------------------------------------------------------*)

procedure RECHERCHERDIMENSION(var ind1,ind2 :integer);

var ch :char;
    trouv,fer,fin : boolean;
    nb : integer;

begin
  nb := 0;
  fer := false;
  trouv := false;
  while not trouv do
    begin
      LECTURECARACTERE(ch,fin);
      if ((ord(ch)>= 48 ) and (ord(ch) <= 57 ))then
        begin
          if nb <> 0 then nb := (ord(ch) - 48) + (nb * 10)
                     else nb := ord(ch) - 48;
        end ;
      if ch = ',' then begin fer := true;ind1 := nb;nb := 0; end;
      if ch = '.' then begin trouv := true;ind2 := nb end;
    end;
  if ch <> '.' then LECPOINT;
end;
```

```
(*****************************************************************************)
(*                PROCEDURES D'IMPLEMENTATION                                *)
(*****************************************************************************)

procedure IMPLEMLISTE(var memmot : pmot;prec : pgeneral;newliste : boolean);

var typel : typeliste;
    i : integer;
    motprec : pmot;

begin
  if  newliste then motprec := memmot;
  new(memmot);
  if not  newliste then
     begin
        prec^.pointmot := memmot;
        typel := prec^.liste;
     end
       else
     begin
       motprec^.pointmot := memmot;
       memmot^.sautligne := true;
       typel := prec^.liste
     end;
     case typel of
        point : memmot^.string[1] := '.';
        tiret : memmot^.string[1] := '-';
        etoile : memmot^.string[1] := '*';
     end;
     memmot^.string[Ø] := chr(1);
     memmot^.fonte := prec^.fonte;
     memmot^.police := prec^.police;
     memmot^.soulignement := prec^.soulignement;
end;
(*****************************************************************************)

procedure NUMEROTATIONENTETE (var memmot : pmot;var tableau : quatreentiers;
                              prec,precp : pgeneral);

begin
                (* reinitialisations *)
  case precp^.niveauchapitre of
    1 : tableau[1] := tableau[1] + 1;
    2 : tableau[2] := tableau[2] + 1;
    3 : tableau[3] := tableau[3] + 1;
    4 : tableau[4] := tableau[4] + 1;
  end;
                (* creation du mot de la numerotation *)
  new(memmot);
  prec^.pointmot := memmot;
  memmot^.fonte := prec^.fonte;
  memmot^.police := prec^.police;
  memmot^.soulignement := prec^.soulignement;
  case precp^.niveauchapitre of
    1 : begin
          memmot^.string[Ø] := chr(3);memmot^.string[1] :=chr(tableau[1]+48);
          memmot^.string[2] := '.';memmot^.string[3] := ' ';
          tableau[2] := Ø;tableau[3] := Ø;tableau[4]:= Ø;
        end;
    2 : begin
          memmot^.string[Ø] :=chr(5);memmot^.string[1]:=chr(tableau[1] + 48);
          memmot^.string[2] := '.';memmot^.string[3] := chr(tableau[2] + 48);
          memmot^.string[4] := '.';memmot^.string[5] := ' ';
          tableau[3] := Ø;tableau[4] := Ø;
        end;
    3 : begin
          memmot^.string[Ø]:= chr(7);memmot^.string[1]:=chr(tableau[1] + 48);

          memmot^.string[2] := '.';memmot^.string[3] := chr(tableau[2] + 48);
          memmot^.string[4] := '.';memmot^.string[5] := chr(tableau[3] + 48);
          memmot^.string[6] := '.';memmot^.string[7] := ' ';
          tableau[4] := Ø;
        end;
    4 : begin
          memmot^.string[Ø]:= chr(9);memmot^.string[1]:=chr(tableau[1] + 48);
          memmot^.string[2]:= '.';memmot^.string[3]:=chr(tableau[2] + 48);
          memmot^.string[4]:= '.';memmot^.string[5]:=chr(tableau[3] + 48);
          memmot^.string[6]:= '.';memmot^.string[7]:=chr(tableau[4] + 48);
          memmot^.string[8]:= '.';memmot^.string[9]:=' ';
        end;
  end; (* fin du case *)
end;
(*****************************************************************************)
```

```
procedure NUMEROTATIONFIG(var memmot : pmot;var numerofig : integer;
                          prec, precp : pgeneral);

var mot,reserve,motcree : pmot;
    i,numero,nbcar,var1 : integer;

begin
  nbcar := Ø; i := 8;numero := numerofig;
  new(memmot);new(motcree);
  prec^.pointmot := memmot;
  memmot^.fonte := gras;
  memmot^.police := prec^.police;
  memmot^.soulignement := prec^.soulignement;
  memmot^.string[1] := '-';     memmot^.string[2] :=' ';
  memmot^.string[3] := 'F';     memmot^.string[4] := 'i';
  memmot^.string[5] :='g';      memmot^.string[6] := '.';
  memmot^.string[7] := ' ';
  mot := precp^.pointmot;
  while mot <> nil do
    begin
      reserve:= mot;
      mot := mot^.pointmot;
    end;
  reserve^.pointmot := motcree;
  motcree^.string[1] := ' ';    motcree^.string[2] := '(';
  motcree^.string[3] := 'f';    motcree^.string[4] := 'i';
  motcree^.string[5] := 'g';    motcree^.string[6] := '.';
  motcree^.string[7] := ' ';
  motcree^.fonte := italique;
  motcree^.police := precp^.police;
  motcree^.soulignement := precp^.soulignement;
  while numero >= 1 do
    begin
      numero := trunc (numero / 1Ø);
      nbcar := nbcar + 1;
    end;
  numero := numerofig;
  memmot^.string[Ø] := chr(nbcar + 1Ø);
  motcree^.string[Ø] := chr(nbcar + 9);
  while nbcar > Ø do
    begin
      var1 := trunc (numero / EXPDIX(nbcar - 1));
      memmot^.string[i] := chr(var1 + 48);
      motcree^.string[i] := chr(var1 + 48);
      i := i + 1;
      numero := numero - (var1 * EXPDIX(nbcar - 1));
      nbcar := nbcar - 1;
    end;
  memmot^.string[i] := ' ';
  motcree^.string[i] := ')';
  memmot^.string[i + 1] := '-';
  motcree^.string[i + 1] := ' ';
  memmot^.string[i + 2] := ' ';
  numerofig := numerofig + 1;
end;
{*******************************************************************************}

procedure NUMEROTATIONNOTE(var memmot : pmot;var numeronote : integer;
                           prec : pgeneral);

var mot,reserve,motcree : pmot;
    i,numero,nbcar,var1 : integer;

begin
  nbcar := Ø; i := 5;numero := numeronote;
  new(memmot);new(motcree);
  prec^.pointmot := memmot;
  memmot^.fonte := prec^.fonte;
  memmot^.police := prec^.police;
  memmot^.soulignement := prec^.soulignement;
  memmot^.string[1] := 'N';     memmot^.string[2] :='B';
  memmot^.string[3] := ':';     memmot^.string[4] := ' ';
  PARCOURSFINNOTE(precp,adrierrecord);
  mot := precp^.pointmot;
  while mot <> nil do
    begin
      reserve:= mot;
      mot := mot^.pointmot;
    end;
  reserve^.pointmot := motcree;
  motcree^.string[1] := ' ';    motcree^.string[2] := '(';
  motcree^.string[3] := 'n';    motcree^.string[4] := 'b';
  motcree^.string[5] := '.';    motcree^.string[6] := ' ';
  motcree^.fonte := italique;
  motcree^.police := precp^.police;
  motcree^.soulignement := precp^.soulignement;
  while numero >= 1 do
    begin
      numero := trunc (numero / 1Ø);
      nbcar := nbcar + 1;
    end;
  numero := numeronote;
  memmot^.string[Ø] := chr(nbcar + 5);
  motcree^.string[Ø] := chr(nbcar + 8);
```

```
  while nbcar > 0 do
    begin
      var1 := trunc (numero / EXPDIX(nbcar - 1));
      memmot^.string[i] := chr(var1 + 48);
      motcree^.string[i+ 2] := chr(var1 + 48);
      i := i + 1;
      numero := numero - (var1 * EXPDIX(nbcar - 1));
      nbcar := nbcar - 1;
    end;
  memmot^.string[i] := ' ';
  motcree^.string[i+ 2] := ')';
  motcree^.string[i + 3] := ' ';
  numeronote := numeronote + 1;
end;
(*****************************************************************************)

procedure LIEN ( precp,prec : pgeneral);

label 3;

var mem,precmem : pgeneral;
    position,i: integer;
    tableau1,tableau2,tableau3,tableau4 : pgeneral;

      procedure parcourschap(pointeur : pgeneral);
        begin
          if pointeur <> nil then
            begin
              if pointeur^.tyfeuille = chapitre then
                begin
                  case pointeur^.niveauchapitre of
                    1 : begin
                          tableau1 := pointeur;
                          tableau2 := nil;
                          tableau3 := nil;
                          tableau4 := nil;
                        end;
                    2 : begin
                          tableau2 := pointeur;
                          tableau3 := nil;
                          tableau4 := nil;
                        end;
                    3 : begin
                          tableau3 := pointeur;
                          tableau4 := nil;
                        end;
                    4 : tableau4 := pointeur;
                  end;
                end;
              parcourschap(pointeur^.pointsuccess1);
              parcourschap(pointeur^.pointsuccess2);
              parcourschap(pointeur^.pointsuivant);
            end;
        end;

begin
  if precp^.tyfeuille = notebaspage then finnotebaspage(precp);
  case prec^.tyfeuille of
    titre        : precp^.pointsuivant    := prec;
    auteur       : precp^.pointsuivant    := prec;
    date         : precp^.pointsuivant    := prec;
    abstract     : precp^.pointsuivant    := prec;
    entete       : precp^.pointsuccess1   := prec;
    paragraphe   : precp^.pointsuivant    := prec;
    figure       : precp^.pointsuivant    := prec;
    liste        : precp^.pointsuivant    := prec;
    notebaspage : begin
                    if precp^.pointsuccess1 = nil
                      then precp^.pointsuccess1 := prec
                      else
                        begin
                          mem := precp^.pointsuccess1;
                          while mem <> nil do
                            begin
                              precmem := mem;
                              mem := mem^.pointsuivant;
                            end;
                          precmem^.pointsuivant := prec;
                        end;
                  end;
    chapitre    : begin
                    tableau1 := nil;tableau2 := nil;
                    tableau3 := nil;tableau4 := nil;
                    parcourschap(adrierrecord);
                    if ((prec^.niveauchapitre = 4 ) and (tableau3 = nil)) then

                        begin
                          ERREUR(202);
                          goto 3;
                        end;
                    if ((prec^.niveauchapitre = 3 ) and (tableau2 = nil)) then

                        begin
                          ERREUR(201);
                          goto 3;
                        end;
                    if ((prec^.niveauchapitre = 2 ) and (tableau1 = nil)) then

                        begin
                          ERREUR(200);
```

```
                        goto 3;
                      end;

                    if tableau4 <> nil then
                      begin
                        case prec^.niveauchapitre of
                          4 : tableau4^.pointsuivant := prec;
                          3 : tableau3^.pointsuivant := prec;
                          2 : tableau2^.pointsuivant := prec;
                          1 : tableau1^.pointsuivant := prec;
                        end;
                      end
                        else
                      begin
                        if tableau3 <> nil then
                          begin
                            case prec^.niveauchapitre of
                              4 : tableau3^.pointsuccess2 := prec;
                              3 : tableau3^.pointsuivant := prec;
                              2 : tableau2^.pointsuivant := prec;
                              1 : tableau1^.pointsuivant := prec;
                            end;
                          end
                            else
                          begin
                            if tableau2 <> nil then
                              begin
                                case prec^.niveauchapitre of
                                  3 : tableau2^.pointsuccess2 := prec;
                                  2 : tableau2^.pointsuivant := prec;
                                  1 : tableau1^.pointsuivant := prec;
                                end;
                              end
                                else
                              begin
                                if tableau1 <> nil then
                                  begin
                                    case prec^.niveauchapitre of
                                      2 : begin
                                            mem := tableau1;
                                            mem^.pointsuccess2 := prec;
                                          end;
                                      1 : tableau1^.pointsuivant := prec;
                                    end;
                                  end
                                   else
                                  precp^.pointsuivant := prec;
                              end;
                          end;
                      end;
                  end;
  end;
  3 : ;
end;

{*****************************************************************************}

procedure CREERRECORD(var ierrecord,prec : pgeneral;var creer : boolean;
                      sorte :typefeuille;ind1,ind2 :integer;precp :pgeneral);
begin
  creer          := true;
  new(prec);
  if sorte = document then ierrecord    := prec;
  case sorte of
    document : begin with prec^ do
                 begin
                   margesuperieure      := margesupdocu;
                   margeinferieure      := margeinfdocu;
                   interentdocu         := interentite;
                   tyfeuille            := sorte;
                   pagination           := paginationdocu;
                   seule                := seulesurpage;
                   superieure           := paginationsuperieure;
                   colonnage            := colonnagedocu;
                   index                := indexdocu;
                   tablematiere         := tablematieredocu;
                   nivtab               := tableniv;
                   poltab               := policetable;
                   fotab                := fontetable;
                   mrgmat               := margegauchetable;
                   mrdmat               := margedroitetable;
                   soutab               := soulitable;
                   centab               := centretable;
                   intertab             := inlitable;
                   impreduite           := impreduite;
                   premiere             := inferieure;
                   derniere             := derniere;
                   tfeuille             := typefeuilledocu;
                   numpage              := numeropage;
                   largeurfeuille       := largeurfeuilledocu;
                   longueurfeuille      := longueurfeuilledocu;
                   pointsuivant         := nil;
                 end;
               end;
```

```
    titre : begin with prec^ do
              begin
                tyfeuille           := sorte;
                centrage            := centretitre;
                margegauche         := margegauchetitre;
                margedroite         := margedroitetitre;
                polan               := policetitre;
                police              := policetitre;
                fonan               := fontetitre;
                fonte               := fontetitre;
                interligne          := inltitre;
                souan               := soultitre;
                soulignement        := soultitre;
                sautligne           := false;
                sautpage            := sauttitre;
                pointsuccess1       := nil;
                pointsuivant        := nil;
                pointmot            := nil;;
              end;
            end;
   auteur : begin with prec^ do
              begin
                tyfeuille           := sorte;
                centrage            := centreauteur;
                margegauche         := margegaucheauteur;
                margedroite         := margedroiteauteur;
                polan               := policeauteur;
                police              := policeauteur;
                fonan               := fonteauteur;
                fonte               := fonteauteur;
                interligne          := inlauteur;
                souan               := soulauteur;
                soulignement        := soulauteur;
                sautligne           := false;
                sautpage            := sautauteur;
                pointsuccess1       := nil;
                pointsuivant        := nil;
                pointmot            := nil;;
              end;
            end;
     date : begin with prec^ do
              begin
                tyfeuille           := sorte;
                centrage            := centredate;
                margegauche         := margegauchedate;
                margedroite         := margedroitedate;
                polan               := policedate;
                police              := policedate;
                fonan               := fontedate;
                fonte               := fontedate;
                interligne          := inldate;
                souan               := souldate;
                soulignement        := souldate;
                sautligne           := false;
                sautpage            := sautdate;
                pointsuccess1       := nil;
                pointsuivant        := nil;
                pointmot            := nil;
              end;
            end;
 abstract : begin with prec^ do
              begin
                tyfeuille           := sorte;
                centrage            := centreresume;
                margegauche         := margegaucheresume;
                margedroite         := margedroiteresume;
                polan               := policeresume;
                police              := policeresume;
                fonan               := fonteresume;
                fonte               := fonteresume;
                interligne          := inlresume;
                souan               := soulresume;
                soulignement        := soulresume;
                sautligne           := false;
                sautpage            := sautresume;
                pointsuccess1       := nil;
                pointsuivant        := nil;
                pointmot            := nil;
              end;
            end;
    liste : begin with prec^ do
              begin
                tyfeuille          := sorte;
                centrage           := centreliste;
                margegauche        := margegaucheliste + decaliste;
                margedroite        := margedroiteliste;
                polan              := policeliste;
                police             := policeliste;
                fonan              := fonteliste;
                fonte              := fonteliste;
                interligne         := inlliste;
                liste              := impliste;
                souan              := soulliste;
                soulignement       := soulliste;
                sautligne          := false;
                sautpage           := sautliste;
                pointsuccess1      := nil;
                pointsuccess2      := nil;
                pointsuivant       := nil;
                pointmot           := nil;;
              end;
            end;
```

```
paragraphe : begin with prec^ do
               begin
                 tyfeuille              := sorte;
                 centrage               := centrepara;
                 margegauche            := margegauchepara;
                 margedroite            := margedroitepara;
                 decalageparagraphe     := decapara;
                 polan                  := policepara;
                 police                 := policepara;
                 fonan                  := fontepara;
                 fonte                  := fontepara;
                 interligne             := inlipara;
                 souan                  := soulipara;
                 soulignement           := soulipara;
                 sautligne              := false;
                 sautpage               := sautparagraphe;
                 pointsuccess1          := nil;
                  pointsuccess2         := nil;
                  pointsuivant          := nil;
                  pointmot              := nil;;
               end;
             end;
    entete : begin with prec^ do
               begin
                 case precp^.niveauchapitre of
                   1 : begin
                         tyfeuille              := sorte;
                         centrage               := centreentniv1;
                         margegauche            := margegaucheentniv1;
                         margedroite            := margedroiteentniv1;;
                         polan                  := policeentniv1;
                         police                 := policeentniv1;
                         fonan                  := fonteentniv1;
                         fonte                  := fonteentniv1;
                         interligne             := inlientniv1;
                         niveauchapitre         := precp^.niveauchapitre;
                         souan                  := soulientniv1;
                         soulignement           := soulientniv1;
                         sautligne              := false;
                         sautpage               := sautentete1;
                         pointsuccess1          := nil;
                         pointsuccess2          := nil;
                         pointsuivant           := nil;
                         pointmot               := nil;
                       end;
                   2 : begin
                         tyfeuille              := sorte;
                         centrage               := centreentniv2;
                         margegauche            := margegaucheentniv2;
                         margedroite            := margedroiteentniv2;;
                         polan                  := policeentniv2;
                         police                 := policeentniv2;
                         fonan                  := fonteentniv2;
                         fonte                  := fonteentniv2;
                         interligne             := inlientniv2;
                         niveauchapitre         := precp^.niveauchapitre;
                         souan                  := soulientniv2;
                         soulignement           := soulientniv2;
                         sautligne              := false;
                         sautpage               := sautentete2;
                         pointsuccess1          := nil;
                         pointsuccess2          := nil;
                         pointsuivant           := nil;
                         pointmot               := nil;
                       end;
                   3 : begin
                         tyfeuille              := sorte;
                         centrage               := centreentniv3;
                         margegauche            := margegaucheentniv3;
                         margedroite            := margedroiteentniv3;;
                         polan                  := policeentniv3;
                         police                 := policeentniv3;
                         fonan                  := fonteentniv3;
                         fonte                  := fonteentniv3;
                         interligne             := inlientniv3;
                         niveauchapitre         := precp^.niveauchapitre;
                         souan                  := soulientniv3;
                         soulignement           := soulientniv3;
                         sautligne              := false;
                         sautpage               := sautentete3;
                         pointsuccess1          := nil;
                         pointsuccess2          := nil;
                         pointsuivant           := nil;
                         pointmot               := nil;
                       end;
                   4 : begin
                         tyfeuille              := sorte;
                         centrage               := centreentniv4;
                         margegauche            := margegaucheentniv4;
                         margedroite            := margedroiteentniv4;;
                         polan                  := policeentniv4;
                         police                 := policeentniv4;
                         fonan                  := fonteentniv4;
                         fonte                  := fonteentniv4;
                         interligne             := inlientniv4;
                         niveauchapitre         := precp^.niveauchapitre;
                         souan                  := soulientniv4;
                         soulignement           := soulientniv4;
                         sautligne              := false;
                         sautpage               := sautentete4;
                         pointsuccess1          := nil;
                         pointsuccess2          := nil;
                         pointsuivant           := nil;
                         pointmot               := nil;
                       end;
                 end;
               end;
             end;
```

```
      chapitre : begin with prec^ do
                   begin
                     tyfeuille              := chapitre;
                     niveauchapitre         := ind1;
                     pointsuccess1          := nil;
                     pointsuccess2          := nil;
                     pointsuivant           := nil;
                   end;
                 end;
        figure : begin with prec^ do
                   begin
                     tyfeuille              := sorte;
                     centrage               := centrefigure;
                     margegauche            := margegauchefigure;
                     margedroite            := margedroitefigure;
                     polan                  := policefigure;
                     police                 := policefigure;
                     fonan                  := fontefigure;
                     fonte                  := fontefigure;
                     interligne             := inlifigure;
                     souan                  := soulifigure;
                     soulignement           := soulifigure;
                     hauteur                := ind1;
                     largeur                := ind2;
                     sautligne              := false;
                     sautpage               := sautfigure;
                     pointsuccess1          := nil;
                     pointsuccess2          := nil;
                     pointsuivant           := nil;
                     pointmot               := nil;
                   end;
                 end;
    notebaspage : begin with prec^ do
                   begin
                     tyfeuille              := sorte;
                     margegauche            := margegauchenote;
                     margedroite            := margedroitenote;
                     polan                  := policenote;
                     police                 := policenote;
                     fonan                  := fontenote;
                     fonte                  := fontenote;
                     interligne             := inlinote;
                     souan                  := soulinote;
                     soulignement           := soulinote;
                     pointsuccess1          := nil;
                     pointsuccess2          := nil;
                     pointsuivant           := nil;
                     pointmot               := nil;
                   end;
                 end;
    end;
  if sorte = document then ierrecord        := prec;
  if precp <> nil then lien(precp,prec);

end;

{*****************************************************************************}
```

```
procedure RECIMPL(var liste,trouveR : boolean;var prec : pgeneral);

var memrec, ch : char;
    tamp : packed array[1..2] of char;
    nb : integer;
    ok ,fin : boolean;

begin
  liste := false;
  ok := false;
  trouver := false;
  LECTURECARACTERE(ch,fin);
  if ch = '\' then
   begin
     liste := true;ok := true;
   end
    else
   begin
    tamp[1] := ch;
    LECTURECARACTERE(ch,fin);
    tamp[2] := ch;
    ch := '#' ;
    case tamp[1] of
      'C','C' : begin
                  case tamp[2] of
                    'r','R' : begin
                                ok := true;
                                trouver := true;
                                prec^.sautligne := true;
                              end;
                    'e','E' : begin
                                prec^.centrage := true;
                                trouver := true;ok := true;
                              end;
                   otherwise ERREUR(203);
                  end;
                  LECPOINT;
                end;
     'd','D' : begin
                 if ((prec^.tyfeuille = titre) or (prec^.tyfeuille = auteur)
                   or (prec^.tyfeuille = abstract) or (prec^.tyfeuille = date)
                   or (prec^.tyfeuille = figure) or (prec^.tyfeuille = entete)
                   or (prec^.tyfeuille = notebaspage)) then  ERREUR(9)
                    else
                   begin
                     LECDEUXPOINT;
                     LECTURECARACTERE(ch,fin);
                     while ch = ' ' do LECTURECARACTERE(ch,fin);
                     DECODAGEVALEUR(nb,ch);
                     if prec^.tyfeuille = paragraphe then
                       begin
                         if nb < 0 then ERREUR(213) else
                                        prec^.decalageparagraphe := nb;
                       end
                         else
                       begin
                         if nb < 0 then ERREUR(220) else
                            prec^.margegauche := prec^.margegauche -
                                 decaliste +  nb;
                       end;
                     ok := true;
                     if ch <> '.' then LECPOINT;
                   end;
               end;
   'f','F' : case TAMP[2] of
               'f','F' : begin (* passage a la page *)
                           ok := true:
                           prec^.sautpage := true;
                           LECPOINT;
                         end;
              'O','o': begin (* changement de fonte *)
                         LECDEUXPOINT;
                         LECTURECARACTERE(ch,fin);
                         while ch = ' ' do LECTURECARACTERE(ch,fin);
                         case ch of
                            'n','N' : begin
                                        ok := true;trouver := true;
                                        prec^.fonte := normal;
                                      end;
                            'i','I' : begin
                                        ok := true;trouver := true;
                                        prec^.fonte := italique;
                                      end;
                            'G','g' : begin
                                        ok := true;trouver := true;
                                        prec^.fonte := gras;
                                      end;
                           otherwise ERREUR(207);
                         end;
                         if ch <> '.' then LECPOINT;
                       end;
              'i','I' : begin
                          LECDEUXPOINT;
                          LECTURECARACTERE(ch,fin);
                          while ch = ' ' do LECTURECARACTERE(ch,fin);
                          case ch of
                            's','S': begin
                                       ok := true;
                                       prec^.soulignement := prec^.souan;
                                       trouver := true;
                                     end;
                            'f','F': begin
                                       ok := true;
                                       prec^.fonte := prec^.fonan;
                                       trouver := true;
```

```
                              'p','P': begin
                                         ok := true;
                                         prec^.police := prec^.polan;
                                         trouver := true;
                                       end;
                                otherwise ERREUR(208);
                              end;
                              if ch <> '.' then LECPOINT;
                            end;
                  otherwise ERREUR(209);
                end;
  'I','i' :  begin
               LECTURECARACTERE(ch,fin);
               LECDEUXPOINT;
               LECTURECARACTERE(CH,fin);
               while ch = ' ' do LECTURECARACTERE(ch,fin);
               case ch of
                 'p','P' : begin
                             ok := true;
                             prec^.liste := point;
                           end;
                 't','T' : begin
                             ok := true;
                             prec^.liste := tiret;
                           end;
                 'E','e' : begin
                             ok := true;
                             prec^.liste := etoile;
                           end;
                 otherwise ERREUR(210);
               end;
             if ch <> '.' then LECPOINT;
           end;
  'j','J' : begin
              prec^.centrage := false;
              trouver := true;ok := true;
              LECPOINT;
            end;
  'l','L' : begin
              LECTURECARACTERE(ch,fin);
              case ch of
                'g','G' : begin
                            prec^.sautligne := true;
                            LECPOINT;
                            ok := true;
                          end;
                otherwise ERREUR(211);
              end;
              if ch <> '.' then LECPOINT;
            end;
  's','S' :  begin
               LECDEUXPOINT;LECTURECARACTERE(ch,fin);
               while ch = ' ' do LECTURECARACTERE(ch,fin);
               case ch of
                 'c','C' : begin
                             ok := true;
                             prec^.soulignement := continu;
                           end;
                 'd','D' : begin
                             ok := true;
                             prec^.soulignement := discontinu;
                           end;
                 'r','R' : begin
                             ok := true;
                             prec^.soulignement := rien;
                           end;
                 otherwise ERREUR(212);
               end;
               if ch <> '.' then LECPOINT
             end;
'm','M' : begin
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do LECTURECARACTERE(ch,fin);
            case ch of
              'g','G' : memrec := 'g';
              'd','D' : memrec := 'd';
              otherwise ERREUR(216);
            end;
            LECDEUXPOINT;
            LECTURECARACTERE(ch,fin);
            while ch = ' ' do LECTURECARACTERE(ch,fin);
            DECODAGEVALEUR(nb,ch);
            case memrec of
              'g' : begin
                      trouver := true;ok := true;
                      if nb < 0 then ERREUR(214) else
                                     prec^.margegauche := nb;
                    end;
              'd' : begin
                      if nb < 0 then ERREUR(215) else
                                     prec^.margedroite := nb + rognage;
                      trouver := true;ok := true;
                    end;
            end;
            if ch <> '.' then LECPOINT;
          end;
  'p','P' : case tamp[2] of
              'a','A' : begin
                          ok := true;
                          prec^.sautpage := true
                        end;
              'o','O' : begin
                          LECDEUXPOINT;
                          LECTURECARACTERE(ch,fin);
```

```
                                nb := -1;ok := true;
                                while ch = ' ' do LECTURECARACTERE(ch,fin);
                                DECODAGEVALEUR(nb,ch);
                                case nb of
                                  10,11,14 : prec^.police := nb;
                                  otherwise ERREUR(205);
                                end;
                                trouver := true;
                                if ch <> '.' then LECPOINT;
                              end;
                  otherwise ERREUR(206);
                end;
    otherwise ERREUR(217);
    end;
  end;
if not ok then
  begin
    write(erreurs,'\');
    write(erreurs,tamp[1]);
    write(erreurs,tamp[2]);
    writeln(erreurs,'          ');
  end;

end;
(*****************************************************************************)

procedure RECLEX (var ch : char; var terrecord,prec : pgeneral;
                  var impl,fanion,trouver,creer, boollliste,
                  carspec : boolean;precp : pgeneral);

var  test : packed array [1..2] of char;
     sorte : typefeuille;
     trouv,bool,ok,pastrouve,newliste  :boolean;
     ind1,ind2,numero : integer;
     memrec : pgeneral;

     procedure CARASPEC(var pastrouve : boolean;var numero : integer;
               ch1,ch2 : char);

     begin
       pastrouve := true;
       case ch1 of
         '''': case ch2 of
                 'e' : numero := 18  ;
                 '\' : numero := 92;
                 'a' : numero := 17;
               otherwise pastrouve := false;
               end;
         '`' : case ch2 of
                 'a' : numero := 16 ;
                 'e' : numero := 19;
                 'u' : numero := 25;
                 '\' : numero := 92;
                 otherwise pastrouve := false;
                 end;
         '^' :case ch2 of
                 'a' : numero := 17;
                 'e' : numero := 21;
                 'i' : numero := 23;
                 'o' : numero := 24;
                 'u' : numero := 26;
                 otherwise pastrouve := false;
                 end;
         '"' :case ch2 of
                 'a' : numero := 27;
                 'e' : numero := 20;
                 'i' : numero := 22;
                 'o' : numero := 28;
                 'u' : numero := 15;
                 otherwise pastrouve := false;
                 end;

         ',' :  if ch2 = 'c' then numero :=29  else pastrouve := false;
         end;
       end;

begin
  carspec := false;      bool := false;          ok := false;
  impl := false;         fanion := true;         newliste := false;
  creer := false;        boollliste := false;
  LECTURECARACTERE(ch,fin);
  if ch = '\' then
    begin
      fanion := false;
      RECIMPL(newliste,trouver,prec);
      impl := true;
      boollliste := newliste ;
    end
     else
    begin
      trouver := false;
      test[1] := ch;
      LECTURECARACTERE(ch,fin);
      test[2] := ch;
```

```
        case test[1] of
          '`','´','^','¨',',' : begin
                                  CARASPEC(pastrouve, numero,test[1],test[2]);
                                  ch := chr(numero);
                                  if not pastrouve then
                                    begin
                                      ERREUR(20) ;
                                      write(erreurs,'\');
                                      write(erreurs,test[1]);
                                      write(erreurs,test[2]);
                                      writeln(erreurs,'          ');
                                    end;
                                  fanion := false;
                                  ok := true;
                                  carspec := true;
                                  trouv := true
                                end;
          'a','A' : begin
                      sorte :=  auteur;ok := true;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true;LECPOINT;
                    end;
          'C','c' : begin
                      sorte := chapitre;ok := true;
                      RECHERCHERNIVEAU(ind1);
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver := true;
                    end;
          'd','D' : begin
                      case test[2] of
                        'o','O' : begin         end;
                        'a','A' : begin
                                    sorte :=  date;
                                    CREERRECORD(ierrecord,prec,creer,sorte,
                                                ind1,ind2,precp);
                                    trouver:=true;LECPOINT;ok := true;
                                  end;
                        otherwise ERREUR(16);
                      end;
                    end;
          'e','E' : begin
                      sorte :=  entete;   ok := true;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true ;LECPOINT;
                    end;
          'f','F' : begin
                      LECTURECARACTERE(ch,fin);
                      case ch of
                        'n','N' : begin
                                    finnotebaspage(prec);

                                    fanion := false;ok := true;
                                              (* cas fin de note bas de page *)
                                  end;
                        'G','g': begin
                                    RECHERCHERDIMENSION(ind1,ind2);


                                    sorte :=  figure;ok := true;
                                    CREERRECORD(ierrecord,prec,creer,sorte,
                                                ind1,ind2,precp);
                                    trouver:=true;
                                  end;
                        otherwise ERREUR(16);
                      end;
                      LECPOINT;
                    end;
          'l','L' : begin
                      sorte :=  liste;ok := true;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true;LECPOINT;
                    end;
          'n','N' : begin
                      sorte :=  notebaspage;ok := true;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true;LECPOINT;
                    end;
          'p','P' : begin
                      sorte :=  paragraphe;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true;ok := true;
                      LECPOINT;
                    end;
          'r','R' : begin
                      sorte :=  abstract;ok := true;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true;LECPOINT;
                    end;
          't','T' : begin
                      sorte := titre ;ok := true;
                      CREERRECORD(ierrecord,prec,creer,sorte,ind1,ind2,precp);
                      trouver:=true;LECPOINT;
                    end;
          'u','U' : begin    end;
          otherwise ERREUR(16);
        end;
    end;
end;

(*****************************************************************************)
```

```
procedure REMPLIRMOT(var newliste,iermot,ierpas,dejasep : boolean;
                     var memmot : pmot;var tableau: quatreentiers;
                     var numeronote,numerofig : integer;
                     precp,prec : pgeneral;ch : char;creer : boolean);

var sortesep : set of char;
    ptrm :pmot;

begin
  sortesep := [';',',','.',' ','!',',','?','+','*','_','-','=','\','/',':','.'
  ,'|',''','<','>'];
  if (iermot or newliste) then
    begin
      case prec^.tyfeuille of
        figure : begin
                   NUMEROTATIONFIG(memmot,numerofig,prec,precp);
                   iermot := false;
                 end;
        notebaspage : begin
                        NUMEROTATIONNOTE(memmot,numeronote,prec);
                        iermot := false;
                      end;
        entete : begin
                   NUMEROTATIONENTETE(memmot,tableau,prec,precp);
                   iermot := false;
                 end;
        liste : begin
                  IMPLEMLISTE(memmot,prec,newliste);
                  iermot := false;
                end;
      end ;
    end;
    if (ch in sortesep) then dejasep := true;
    if ((ierpas or ((not(ch  in sortesep)) and dejasep )) or creer
        or prec^.sautligne) then
      begin
        (* test pour voir si on a un chapitre avec un mot avant l'entete*)
        if prec^.tyfeuille = chapitre then ERREUR(8);
        creer := false;
        new(ptrm);
        ptrm^.string[0]:=chr(1);
        ptrm^.string[1] := ch;
        if iermot then
          begin
            prec^.pointmot := ptrm;
            ptrm^.pointmot := nil;
            memmot := ptrm;
            iermot := false;
          end
            else
          begin
            ptrm^.pointmot := nil;
            memmot^.pointmot := ptrm;
            memmot := ptrm;
          end;
        ptrm^.soulignement := prec^.soulignement;
        ptrm^.fonte := prec^.fonte;
        ptrm^.police := prec^.police;
        ptrm^.sautligne := prec^.sautligne;
        prec^.sautligne := false;
        if (not (ch in sortesep)) then dejasep := false else dejasep := true;
        ierpas := false
      end
        else
      begin
        memmot^.string[0] := chr(ord(memmot^.string[0])+  1);
        memmot^.string[ord(memmot^.string[0])] :=ch;
      end;
end;
```

```
(*****************************************************************************)
(*                        PROCEDURE ARBRE STRUCTURE - CORPS                  *)
(*****************************************************************************)

begin
  precp := nil;
  fermot := true;
  ferpas := true;
  dejasep := false;
  reset(source);
  DEFELEMBASE;  (*definition des elements de base*)
TEST1;
writeln(res,'-----------------------------------------');
  MODELEMBASE;  (* modification des elements de base*)
TEST1;
  precp := nil;
  tableau[1] := numeroentete;
  for i := 2 to 4 do tableau[i] := Ø;
  ind1 := Ø ; ind2 := Ø;
  notdejaslash := true;
  CREERRECORD(adrferrecord,prec,creer,document,ind1,ind2,precp);

  LECTURECARACTERE(ch,fin);
  precp := prec;
  ferrecord := adrferrecord;
  while not eof(source) do
    begin
      while not eoln(source) do
        begin
          if (ch = '\') then
            begin
              memnewliste := newliste;
              RECLEX(ch,ferrecord,prec,impl,fanion,trouver,creer,newliste,
                     carspec,precp);
              if carspec then newliste := memnewliste;
              precpp := precp;
              creerliste := false;
              if not fermot then
                begin
                if prec^.soulignement <> memmot^.soulignement
                        then creerliste := true;
                end;
         (* au cas ou l'on a un changement de soulignement meme si l'on
          est au sein d'un mot, on doit forcer la creation d'un nouveau mot *)
              if (not impl and not carspec)then precp := prec;
              if (carspec and fermot) then fanion := true;
              if fanion then
                begin
                  fermot := true;
                  ferpas := true;
                  memmot := nil;
                  dejasep := false;
                end
            end
            else
            begin
              REMPLIRMOT (newliste,fermot,ferpas,dejasep,memmot,tableau,
                          numeronote,numerofig,precpp,prec,ch,creerliste);
              creerliste := false;
              carspec := false;
              newliste := false;
            end;
        (* au cas ou l'on a eu un separateur, on doit forcer le traitement
              de ce caractere *)
            if carspec then
              begin
                REMPLIRMOT (newliste,fermot,ferpas,dejasep,memmot,tableau,
                            numeronote,numerofig,precpp,prec,ch,creerliste);
                creerliste := false;
                carspec := false;
                newliste := false;
              end;
            read(source,ch);
          end;
        readln(source);
      end;
    REMPLIRMOT (newliste,fermot,ferpas,dejasep,memmot,tableau,
        numeronote,numerofig,precpp,prec,ch,creerliste);
end;
(*****************************************************************************)
```

```
(*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)
(*::::::::::::::::::::ARBRE DE FORMATAGE::::::::::::::::::::::::::::::::::::::::::::::*)
(*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)

procedure ARBREFORMATAGE( var pointpage : ptypographique; pointdoc : PGENERAL);


var arbfor, colonnagetypo,indextypo,tablematieretypo : boolean;
    pointentite,mementite,adrpage,mempage,iertab : ptypographique;
    tableau11g,tableau11i,tableau11r,tableau10r,
    tableau14 : array[1..3,0..127] of integer;
    ecartmoyen : integer;
{*****************************************************************************}

procedure BINAIREDECIMAL( var entier : integer;byy : byte);

begin
  entier := 0;
  if byy[0] = 1 then entier := entier + 128 ;
  if byy[1] = 1 then entier := entier + 64 ;
  if byy[2] = 1 then entier := entier + 32;
  if byy[3] = 1 then entier := entier + 16;
  if byy[4] = 1 then entier := entier + 8;
  if byy[5] = 1 then entier := entier + 4;
  if byy[6] = 1 then entier := entier + 2;
  if byy[7] = 1 then entier := entier + 1;
end;
{*****************************************************************************}


procedure TABR10;

var val,compteur,position,indice1,place ,valeur: integer;
    cara : char;
    largeur : byte;

begin
  compteur := 0;
  position := 0;
  indice1 := 1;
  place := 0;
  reset(r10);
  while not eof(r10) do
    begin
      while not eoln(r10) do
        begin
          read(r10,cara);
          compteur := compteur + 1;
          if (compteur >= ((256 * 8) + 1))  then
            begin
              if cara = '1' then val := 1 else val := 0;
              largeur[position] := val;
              position := position + 1;
              if position > 7 then
                begin
                  BINAIREDECIMAL(valeur,largeur);
                  tableau10r[indice1,place] := valeur;
                  place := place + 1;
                  position := 0;
                  if place > 127 then
                    begin
                      indice1 := indice1 + 1;
                      place := 0;
                    end;
                end;
            end;
        end;
      readln(r10);
    end;
end;
{*****************************************************************************}


procedure TABG11;

var val,compteur,position,indice1,place ,valeur: integer;
```

```
    cara : char;
    largeur : byte;

begin
  compteur := Ø;
  position := Ø;
  indicel := 1;
  place := Ø;
  reset(g11);
  while not eof(g11) do
    begin
      while not eoln(g11) do
        begin
          read(g11,cara);
          compteur := compteur + 1;
          if (compteur >= ((256 * 8) + 1))  then
            begin
              if cara = '1' then val := 1 else val := Ø;
              largeur[position] := val;
              position := position + 1;
              if position > 7 then
                begin
                  BINAIREDECIMAL(valeur,largeur);
                  tableaullg[indicel,place] := valeur;
                  place := place + 1;
                  position := Ø;
                  if place > 127 then
                    begin
                      indicel := indicel + 1;
                      place := Ø;
                    end;
                end;
            end;
        end;
      readln(g11);
    end;
end;
(*****************************************************************************)

procedure TABI11;

var val,compteur,position,indicel,place ,valeur: integer;
    cara : char;
    largeur : byte;

begin
  compteur := Ø;
  position := Ø;
  indicel := 1;
  place := Ø;
  reset(i11);
  while not eof(i11) do
    begin
      while not eoln(i11) do
        begin
          read(i11,cara);
          compteur := compteur + 1;
          if (compteur >= ((256 * 8) + 1))  then
            begin
              if cara = '1' then val := 1 else val := Ø;
              largeur[position] := val;
              position := position + 1;
              if position > 7 then
                begin
                  BINAIREDECIMAL(valeur,largeur);
                  tableaulli[indicel,place] := valeur;
                  place := place + 1;
                  position := Ø;
                  if place > 127 then
                    begin

                      indicel := indicel + 1;
                      place := Ø;
                    end;
                end;
            end;
        end;
      readln(i11);
    end;
end;
(*****************************************************************************)

procedure TABR11;

var val,compteur,position,indicel,place ,valeur: integer;
    cara : char;
    largeur : byte;
```

```
 begin
   compteur := Ø;
   position := Ø;
   indice1 := 1;
   place := Ø;
   reset(r11);
   while not eof(r11) do
     begin
       while not eoln(r11) do
         begin
           read(r11,cara);
           compteur := compteur + 1;
           if (compteur >= ((256 * 8) + 1))  then
             begin
               if cara = '1' then val := 1 else val := Ø;
               largeur[position] := val;
               position := position + 1;
               if position > 7 then
                 begin
                   BINAIREDECIMAL(valeur,largeur);
                   tableau11r[indice1,place] := valeur;
                   place := place + 1;
                   position := Ø;
                   if place > 127 then
                     begin
                       indice1 := indice1 + 1;
                       place := Ø;
                     end;
                 end;
           end;
         end;
       readln(r11);
     end;
end;

(*****************************************************************************)

procedure TAB14;

var val,compteur,position,indice1,place ,valeur: integer;
    cara : char;
    largeur : byte;

begin
  compteur := Ø;
  position := Ø;
  indice1 := 1;
  place := Ø;
  reset(r14);
  while not eof(r14) do
    begin
      while not eoln(r14) do
        begin
          read(r14,cara);
          compteur := compteur + 1;
          if (compteur >= ((256 * 8) + 1))  then
            begin
              if cara = '1' then val := 1 else val := Ø;
              largeur[position] := val;
              position := position + 1;
              if position > 7 then
                begin
                  BINAIREDECIMAL(valeur,largeur);
                  tableau14[indice1,place] := valeur;
                  place := place + 1;
                  position := Ø;
                  if place > 127 then
                    begin
                      indice1 := indice1 + 1;
                      place := Ø;
                    end;
                end;
          end;
        end;
      readln(r14);
    end
end;
```

```
(*****************************************************************************)

procedure EXAMEN(var tphauteur,tplargeur : integer; tplettre : char;
                     tyfont : typefonte;tychar : integer);

var val : integer;

begin
  val := ord (tplettre);
  case tychar of
    10 : begin
           tplargeur := tableau10r[1,val];
           tphauteur := tableau10r[2,val] + tableau10r[3,val];
         end ;
    11 : case tyfont of
           normal : begin
                      tplargeur := tableau11r[1,val];
                      tphauteur := tableau11r[2,val] + tableau11r[3,val];
                    end ;
           italique : begin
                        tplargeur := tableau11i[1,val];
                        tphauteur := tableau11i[2,val] + tableau11i[3,val];
                      end ;
           gras : begin
                    tplargeur := tableau11g[1,val];
                    tphauteur := tableau11g[2,val] + tableau11g[3,val];
                  end ;
         end;
    14 : begin
           tplargeur := tableau14[1,val];
           tphauteur := tableau14[2,val] + tableau14[3,val];
         end ;
  end;
  if ((tychar <> 11 ) or (tychar <> 10 ) or (tychar <> 14)) then
    begin
      case tyfont of
       normal : begin
                  tplargeur := tableau11r[1,val];
                  tphauteur := tableau11r[2,val] + tableau11r[3,val];
                end ;
       italique : begin
                    tplargeur := tableau11i[1,val];
                    tphauteur := tableau11i[2,val] + tableau11i[3,val];
                  end ;
       gras : begin

                tplargeur := tableau11g[1,val];
                tphauteur := tableau11g[2,val] + tableau11g[3,val];
              end;
       end;
  end;
end;

(*****************************************************************************)

procedure INIT(var nbcar,position,hauteurmot,largeurmot,compteurblancmot,
        compteurblancboutli,compteurblancli,hauteurboutli,hauteurli,largeurli,
        largeurboutli,memlargeurli,memlargeurboutli,memhauteurboutli,nbli,
        nbboutli,reste,adresseboutli,positionboutligne,hauteuren : integer);

begin
  nbcar                := 0;    position             := 0;
  hauteurmot           := 0;    largeurmot           := 0;
  compteurblancmot     := 0;    compteurblancboutli  := 0;
  compteurblancli      := 0;    hauteurboutli        := 0;
  hauteurli            := 0;    largeurli            := 0;
  largeurboutli        := 0;    memlargeurli         := 0;
  memlargeurboutli     := 0;    memhauteurboutli     := 0;
  nbli                 := 0;    nbboutli             := 0;
  hauteuren            := 0;    reste                := 0;
  adresseboutli        := 0;    positionboutligne    := 0;
end;

(*****************************************************************************)

procedure CREERRECORD(var ptm : ptypographique;po : integer;fo :typefonte;
                  sou : typesoulignement;sorte : tyfor;nbpointentite : integer);

begin
  new(ptm);
  ptm^.typo                        := sorte;
  ptm^.x                           := 0;
  ptm^.y                           := 0;
  ptm^.deltax                      := 0;
  ptm^.deltay                      := 0;
  ptm^.policetypographique         := po;
  ptm^.fontetypographique          := fo;
  ptm^.soulignementtypographique   := sou;
  ptm^.pointsuccess                := nil;
  ptm^.pointsuivant                := nil;
  ptm^.valuespace                  := 0;
  ptm^.nombreelement               := 0;
  ptm^.nombreblanc                 := 0;
  ptm^.largeurvoulue               := nbpointentite;
end;

(*****************************************************************************)
```

```
procedure GERERRECORD(gx,gy,gdeltax,gdeltay,gnombreelem,gspace,gblanc :integer;
                      ptm : ptypographique;sorte :tyfor);

begin
  ptm^.x              := gx;              ptm^.y              := gy;
  ptm^.deltax         := gdeltax;         ptm^.deltay         := gdeltay;
  ptm^.nombreelement  := gnombreelem;     ptm^.valuespace     := gspace;
  ptm^.nombreblanc    := gblanc;
end;

{*************************************************************************}

procedure JUSTIFIE (var long :integer;largblanc :integer;
                    pligne :ptypographique;compteurblancli :integer);

var cara,ligneutiliseesansblanc,reste ,compt,largcara: integer;
    inter : ptypographique;

begin
  cara := 0;
  compt := 0;
  inter := pligne^.pointsuccess;
  while inter   <> nil do
    begin
      cara := cara + inter^.nombreelement;
      compt := compt + inter^.deltax;
      inter  :=   inter^.pointsuivant;
    end;
  ligneutiliseesansblanc := compt - (largblanc * compteurblancli);
  reste := pligne^.largeurvoulue - ligneutiliseesansblanc;
  if compteurblancli <> 0 then long := trunc(reste / compteurblancli)
                          else long := 0;
  if cara <> 0 then
    begin
      if long > ((ligneutiliseesansblanc div cara ) * 3)
        then long := ((ligneutiliseesansblanc div cara) * 3);
    end
    else long := largblanc;
end;

{*************************************************************************}

procedure TRTCENTRAGE(pentite : ptypographique;nbpointentite,margegauche,
                      margedroite : integer);

var pli,pboutli : ptypographique;
    travail1,travail2,decalage : integer;

begin
  pli := pentite^.pointsuccess;
  while pli <> nil do
    begin
      travail1 := nbpointentite - margegauche - margedroite;
      travail2 := pli^.deltax;
      if (travail1 - travail2) > 0 then
        decalage := round((travail1 - travail2) / 2);
      pli^.x := pli^.x + decalage;
      pboutli := pli^.pointsuccess;
      while pboutli <> nil do
        begin
          pboutli^.x := pboutli^.x + decalage;
          pboutli := pboutli^.pointsuivant;
        end;
      pli := pli^.pointsuivant;
    end;
end;

{*************************************************************************}

procedure MISEAJOUR(var hautbout,hautli,largli,largbout,hautmot,
                    largmot : integer);

begin
  if hautbout < hautmot then hautbout := hautmot;
  if hautli < hautmot then hautli := hautmot;
  largli := largli + largmot;
  largbout := largbout + largmot;
end;

{*************************************************************************}

procedure TRTMOT(var nbcar,position,hauteurmot,largeurmot,compteurblancmot :
                 integer;var tampon : ensemble;pointmot : pmot;fo : typefonte;
                 po : integer;largeurblanc : integer);
```

```
var i,haut,larg : integer;
    tplettre : char;

begin
  i := 1;
  hauteurmot := Ø;
  largeurmot := Ø;
  compteurblancmot := Ø;
  while i <= ord(pointmot^.string[Ø]) do
    begin
      position := position + 1;
      tplettre := pointmot^.string[i];
      tampon[position] := tplettre;
      if tplettre = ' ' then
        begin
          compteurblancmot := compteurblancmot + 1;
          largeurmot := largeurmot + largeurblanc;
        end
        else
        begin
          EXAMEN(haut,larg,tplettre,fo,po);
          largeurmot := largeurmot + larg;
        if ((tplettre <> '{') or (tplettre <> '[')or (tplettre <> '(')
          or (tplettre <> '}')or (tplettre <> ']') or (tplettre <> ')')) then
          begin
            if hauteurmot < haut then hauteurmot := haut;
          end;
      end;
    i := i + 1;
  end;
  nbcar := ord(pointmot^.string[Ø]);
end;
{*****************************************************************************}

procedure TRTFINLIGNE(var largeurblanc,largeurli,hauteurli,compteurblancli :
                          integer;pboutligne,pligne : ptypographique;
                          largblanc,memlargeurli : integer);

var largeuravecblanc,largeursansblanc,nombreelementavecblanc,difference,
        nombreelementsansblanc,caralargmoyen,compteurblancboutli :integer;
        pboutli : ptypographique;

begin
  largeuravecblanc := Ø;
  nombreelementavecblanc := Ø;
  pboutli := pligne^.pointsuccess;
  while pboutli <> nil do
    begin
      largeuravecblanc := largeuravecblanc + pboutligne^.deltax;
      nombreelementavecblanc := nombreelementavecblanc +
                                pboutli^.nombreelement;
      pboutli := pboutli^.pointsuivant;
    end;
  largeursansblanc := largeuravecblanc - (compteurblancli * largblanc);
  nombreelementsansblanc := nombreelementavecblanc - compteurblancli;
  if nombreelementsansblanc <> Ø
    then caralargmoyen := largeursansblanc div nombreelementsansblanc
    else caralargmoyen := largeuravecblanc;
  if caralargmoyen < largblanc
    then largeurblanc := largblanc
    else largeurblanc := caralargmoyen;
  largeurli := largeurli + memlargeurli;
  largeurli := ((largeurblanc - largblanc )* compteurblancli) + largeurli;


        (* ajustement au cas ou l'on a un depassement de la ligne *)

  if largeurli > pligne^.largeurvoulue then
    begin
      difference := largeurli - pligne^.largeurvoulue;
      difference := round(difference / compteurblancli);
      largeurblanc := largeurblanc - difference;
      largeurli := largeurli - (difference * compteurblancli);
    end;

        (* gestion des bouts de ligne *)

  hauteurli := Ø;
  pboutli := pligne^.pointsuccess;
  while pboutli <> nil do
    begin
      if hauteurli < pboutli^.deltay then hauteurli := pboutli^.deltay;
      compteurblancboutli := pboutli^.nombreblanc;
      pboutli^.valuespace := largeurblanc;
      pboutli^.deltax := pboutli^.deltax + ((largeurblanc - largblanc)
                         * pboutli^.nombreblanc);
      pboutli := pboutli^.pointsuivant;
    end;
end;

{*****************************************************************************}
```

```
procedure TRAITEMENTCHANGEMENT(
        var pointmot : pmot;
        var reste,adresseboutli,position,memlargeurboutli,largblanc,
            memlargeurli,compteurblancboutli,compteurblancli,largeurblanc,
            positionboutligne,memhauteurboutli,nbboutli,nbli,largeurboutli,
            largeurli,nbpointentite,hauteuren,hauteurboutli,hauteurli,
            interlignetypo,compteurblanc          : integer;
        var pligne,pboutligne : ptypographique;
        var centragetypo : boolean;
        var tampon : ensemble;
        fonte : typefonte;
        soulignement : typesoulignement;
        police,nblettreboutli,margegauchetypo : integer);

var i,j,pos : integer;
    sorte : tyfor;
    lettre : char;
    tphaut,tplar : integer;
    memligne,memboutli : ptypographique;

begin
  if ((reste < 0)  or pointmot^.sautligne) then
    begin
      pointmot^.sautligne := false;
      i := 1;
      while i <= adresseboutli  do
        begin
          pboutligne^.stringtypographique[i] := tampon[i];
          i := i + 1;
        end;
      positionboutligne := i - 1;

        (* procedure retirant les blanc en fin de ligne *)

      while pboutligne^.stringtypographique[positionboutligne] = ' ' do
        begin
          positionboutligne := positionboutligne - 1;
          memlargeurboutli := memlargeurboutli - largblanc;
          memlargeurli := memlargeurli - largblanc;
          compteurblancboutli := compteurblancboutli - 1;
          compteurblancli := compteurblancli -1;
        end;
      pboutligne^.nombreelement := positionboutligne;
      i := 1;
      while (adresseboutli + 1) <= position do
        begin
          tampon[i] := tampon[adresseboutli + 1];
          i := i + 1;

          adresseboutli := adresseboutli + 1;
        end;
      position := i ;
      pboutligne^.deltax := memlargeurboutli;
      if not pointmot^.sautligne then
        JUSTIFIE(largeurblanc,largblanc,pligne,compteurblancli);
      sorte := boutlignetypo;
      GERERRECORD(0,0,memlargeurboutli,memhauteurboutli,pboutligne^.nombreelemen
t,
                  largeurblanc,compteurblancboutli,pboutligne,sorte);
      sorte := boutlignetypo;
      CREERRECORD(pboutligne,pointmot^.police,pointmot^.fonte,
                  pointmot^.soulignement,sorte,nbpointentite);
      hauteurli := 0;

        (* petite procedure, necessite de modifier le x des boutli et la
                valeur de leur largeurblanc *)

      memboutli := pligne^.pointsuccess;
      pos := memboutli^.x;
      while memboutli <> nil do
        begin
          memboutli^.valuespace := largeurblanc;
          memboutli^.deltax := memboutli^.deltax + (memboutli^.nombreblanc *
                               (largeurblanc - largblanc));
          memboutli^.x := pos;
          if hauteurli < memboutli^.deltay then hauteurli := memboutli^.deltay;

          pos := pos + memboutli^.deltax ;
          memboutli := memboutli^.pointsuivant;
        end;
      sorte := lignetypo;
      GERERRECORD(margegauchetypo,0,memlargeurli + (compteurblancli *
                  (largeurblanc - largblanc)),hauteurli,nbboutli,
                  largeurblanc,compteurblancli,pligne,sorte);
      lettre := 'i';
      EXAMEN (tphaut,tplar,lettre,pointmot^.fonte,pointmot^.police);
      largblanc := round(tplar * proportionblanc);
      memligne := pligne;
      sorte := lignetypo;
      CREERRECORD(pligne,pointmot^.police,pointmot^.fonte,
                  pointmot^.soulignement,sorte,nbpointentite);
```

```
      memligne^.pointsuivant := pligne;
      pligne^.pointsuccess := pboutligne;
      nbli                 := nbli + 1;
      nbboutli             := 1;
      positionboutligne    := Ø;
      position             := Ø;
      largeurboutli        := Ø;
      largeurli            := Ø;
      memlargeurli         := Ø;
      reste                := nbpointentite;
      memlargeurboutli     := Ø;
      hauteuren            := hauteuren + interlignetypo + hauteurli;
      hauteurli            := Ø;
      memhauteurboutli     := Ø;
      hauteurboutli        := Ø;
      compteurblancli      := Ø;
      compteurblancboutli  := Ø;
      compteurblanc        := Ø;
    end
      else
    begin
      if ((pointmot^.fonte <> pboutligne^.fontetypographique) or
         (pointmot^.soulignement <> pboutligne^.soulignementtypographique) or
         (pointmot^.police <> pboutligne^.policetypographique)) then
        begin
          sorte := boutlignetypo;
          memboutli := pboutligne;
          GERERRECORD(Ø,Ø,memlargeurboutli,memhauteurboutli,position -
                nblettreboutli,largblanc,compteurblancboutli,pboutligne,sorte);

          for i := 1 to  (position - nblettreboutli) do
                        pboutligne^.stringtypographique[i] := tampon[i];
          j := 1;
          for i := (position - nblettreboutli + 1) to position do
            begin
              tampon[j]  := tampon[i];
              j := j + 1;
            end;
          position := j - 1;
          CREERRECORD( pboutligne,pointmot^.police,pointmot^.fonte,
                        pointmot^.soulignement,sorte,nbpointentite);
          memboutli^.pointsuivant := pboutligne;
          nbboutli := nbboutli + 1;
          memlargeurboutli := Ø;
          memhauteurboutli := Ø;
          compteurblancboutli := Ø;
        end;
       pointmot := pointmot^.pointmot;
       if pointmot = nil then
         begin
           i := 1;
           positionboutligne := Ø;
           pboutligne^.nombreelement := position;
           while i <= position do
             begin
               positionboutligne := positionboutligne + 1;
               pboutligne^.stringtypographique[positionboutligne] := tampon[i];
               i := i + 1;
             end;
           sorte := boutlignetypo;
           GERERRECORD (Ø,Ø,largeurboutli + memlargeurboutli,hauteurboutli,
                position,Ø,compteurblancboutli + compteurblanc,pboutligne,sorte);
           compteurblancli := compteurblancli + compteurblanc;
           TRTFINLIGNE(largeurblanc,largeurli,hauteurli,compteurblancli ,
                        pboutligne,pligne,largblanc,memlargeurli);
         end;
    end;
end;

{*****************************************************************************}

procedure TRTCOMPLETAGE(pentite : ptypographique;hauteuren,
                        interlignetypo : integer);

var pos,valx,variable,nbbout,accroit : integer;
    pboutligne,pligne : ptypographique;

begin
  pos := hauteuren;
  pligne := pentite^.pointsuccess;
  while pligne <> nil do
    begin
      variable := pligne^.largeurvoulue - pligne^.deltax;
      pos := pos - pligne^.deltay;
      pligne^.y := pos;
      pboutligne := pligne^.pointsuccess;
      valx := pligne^.x;
      while pboutligne <> nil do
        begin
          nbbout := pboutligne^.nombreblanc;
          if (nbbout - variable) > Ø then
            begin
              accroit := variable;
              variable := Ø;
            end
             else
            begin
              accroit := nbbout;
              variable := variable - nbbout;
            end;
```

```
          pboutligne^.x := valx;
          pboutligne^.y := pos;
          valx := valx + pboutligne^.deltax + accroit;
          pboutligne := pboutligne^.pointsuivant;
        end;
      pos := pos - interlignetypo;
      pligne := pligne^.pointsuivant;
    end;
end;

(*****************************************************************************)

procedure TRAITEMENTCLASSIQUE (
        var pointen : ptypographique;pointmot : pmot;
        nbpointentite,margegauchetypo,margedroitetypo,policetypo,
        interlignetypo,decalage : integer;
        ierligne,centragetypo : boolean;
        fontetypo : typefonte;
        soulignementtypo : typesoulignement);


var nbcar,postion,hauteurmot,largeurmot,compteurblanc,compteurblancboutli,
    compteurblancli,hauteurboutli,hauteurli,largeurli,largeurboutli,
    memlargeurli,memlargeurboutli,largeurblanc,memhauteurboutli,nbli,
    hauteuren,largeurlistandart,reste,haut,larg,largblanc,posdebmot,nbboutli,
    adresseboutli,positionboutli,memnbpointentite,position  : integer;
    tampon : ensemble;
    pentite,pligne,pboutligne : ptypographique;
    lettre : char;
    sorte : tyfor;

begin
  INIT(nbcar,position,hauteurmot,largeurmot,compteurblanc,compteurblancboutli,
IB      compteurblancli,hauteurboutli,hauteurli,largeurli,largeurboutli,
        memlargeurli,memlargeurboutli,memhauteurboutli,nbli,nbboutli,reste,
        adresseboutli,positionboutli,hauteuren);
  sorte := entitetypo;
  CREERRECORD(pentite,pointmot^.police,pointmot^.fonte,
              pointmot^.soulignement,sorte,nbpointentite);
  sorte := lignetypo;
  CREERRECORD(pligne,pointmot^.police,pointmot^.fonte,
              pointmot^.soulignement,sorte,nbpointentite - decalage);
  pentite^.pointsuccess := pligne;
  nbli := nbli + 1;
  sorte := boutlignetypo;
  CREERRECORD(pboutligne,pointmot^.police,pointmot^.fonte,
              pointmot^.soulignement,sorte,nbpointentite - decalage);
  pligne^.pointsuccess := pboutligne;
  nbboutli := nbboutli + 1;
  reste := nbpointentite;

        (* initialisation de la largeur minimal du blanc a priori *)

  lettre := 'i';
  EXAMEN(haut,larg,lettre,pointmot^.fonte,pointmot^.police);
  largblanc := round(larg * proportionblanc);
  while pointmot <> nil do
    begin
      adresseboutli := position;
      memlargeurli := largeurli + memlargeurli;
      memlargeurboutli := largeurboutli + memlargeurboutli;
      memhauteurboutli := 0;
      largeurli := 0;
      largeurboutli := 0;
      if hauteurboutli > memhauteurboutli then memhauteurboutli :=
                                               hauteurboutli;
      compteurblancli := compteurblancli + compteurblanc;
      compteurblancboutli := compteurblancboutli + compteurblanc;
      posdebmot := position + 1;
      if ierligne then
        begin
          memnbpointentite := nbpointentite;
          nbpointentite := nbpointentite - decalage;
          ierligne := false;
        end
        else
        nbpointentite := memnbpointentite;
      TRTmot(nbcar,position ,hauteurmot,largeurmot,compteurblanc,tampon,
             pointmot,pointmot^.fonte,pointmot^.police,largblanc);
      MISEAJOUR (hauteurboutli,hauteurli,largeurli,largeurboutli,
                 hauteurmot,largeurmot);
      reste := reste - largeurmot;
      TRAITEMENTCHANGEMENT(pointmot,reste,adresseboutli,position,
                memlargeurboutli,largblanc,memlargeurli,compteurblancboutli,
                compteurblancli,largeurblanc,positionboutli,memhauteurboutli,
                nbboutli,nbli,largeurboutli,largeurli,nbpointentite,hauteuren,
                hauteurboutli,hauteurli,interlignetypo,compteurblanc,pligne,
                pboutligne,centragetypo,tampon,fontetypo,soulignementtypo,
                policetypo,nbcar,margegauchetypo);
  end;
  sorte := lignetypo;
  GERERRECORD(margegauchetypo,0,largeurli,hauteurli,nbboutli,largeurblanc,
              compteurblancli,pligne,sorte);
  hauteuren := hauteuren + hauteurli;
  TRTCOMPLETAGE(pentite,hauteuren,interlignetypo);
  pligne := pentite^.pointsuccess;
  pligne^.x := pligne^.x + decalage;
  pboutligne := pligne^.pointsuccess;
  while pboutligne <> nil do
    begin
      pboutligne^.x:= pboutligne^.x + decalage;
      pboutligne := pboutligne^.pointsuivant;
    end;
```

```
  if centragetypo then
    TRTCENTRAGE(pentite,nbpointentite,margegauchetypo,margedroitetypo);
  sorte := entitetypo;
  GERERRECORD(margegauchetypo,Ø,nbpointentite,hauteuren,nbll,Ø,Ø,pentite,sorte);

  pointen := pentite;
end;
{*****************************************************************************}

procedure TRTARBREFORMATAGE(var mementite,pointentite : ptypographique;
                            var arbfor,colonnage : boolean;
                            larpoipa : integer;pointdocument : pgeneral);

var  margegauchetypo,margedroitetypo,interlignetypo,nbpointentite,
     hauteurtypo,largeurtypo,decalagetypo : integer;
     centragetypo,feuilletypo,ierligne : boolean;
     fontetypo : typefonte;
     listetypo : typeliste;
     soulignementtypo :typesoulignement;
     typefeuilletypo : typefeuille;
     policetypo : 1..2Ø;
     valeurentite,figentite : ptypographique;

begin
  if pointdocument <> nil then
    begin
      if pointdocument^.pointmot <> nil then
        begin
          margegauchetypo        := pointdocument^.margegauche;
          margedroitetypo        := pointdocument^.margedroite;
          policetypo             := pointdocument^.police;
          interlignetypo         := pointdocument^.interligne;
          centragetypo           := pointdocument^.centrage;
          fontetypo              := pointdocument^.fonte;
          listetypo              := pointdocument^.liste;
          soulignementtypo       := pointdocument^.soulignement;
          typefeuilletypo        := pointdocument^.tyfeuille;
          hauteurtypo            := pointdocument^.hauteur;
          largeurtypo            := pointdocument^.largeur;
          if pointdocument^.tyfeuille = paragraphe
           then decalagetypo :=
                 round(nbpointmm * pointdocument^.decalageparagraphe)
           else decalagetypo := Ø;
          ierligne := true;
          if centragetypo then
            begin
              margegauchetypo := Ø;
              margedroitetypo := rognage;
            end;
         margegauchetypo := trunc(nbpointmm * margegauchetypo);
         margedroitetypo := trunc(nbpointmm * margedroitetypo);
         nbpointentite := larpoipa - margegauchetypo - margedroitetypo;
         if (colonnage and (pointdocument^.tyfeuille <> notebaspage)) then
           begin
            nbpointentite := (nbpointentite div 2);
            margegauchetypo := margegauchetypo div 2;
            margedroitetypo := margedroitetypo div 2;
            if margedroitetypo < 2Ø then margedroitetypo := 2Ø;
           end;
        valeurentite := pointentite;
        interlignetypo := trunc(nbpointmm *      interlignetypo );
        if pointdocument^.tyfeuille <> chapitre then
            TRAITEMENTCLASSIQUE(pointentite,pointdocument^.pointmot,
                        nbpointentite,margegauchetypo,margedroitetypo,
                        policetypo,interlignetypo,decalagetypo,ierligne,
                        centragetypo,fontetypo,soulignementtypo);
        if pointdocument^.tyfeuille = liste then pointentite^.x := Ø;
        pointentite^.sautpagetypographique := pointdocument^.sautpage;
        pointentite^.interligneentite := interlignetypo ;
        pointentite^.typesorte := pointdocument^.tyfeuille;
        if pointdocument^.tyfeuille = figure then
          begin
            new(figentite);
            figentite^.typo := entitetypo;
            figentite^.typesorte := figure;
            figentite^.deltax := trunc(hauteurtypo * nbpointmm);
            figentite^.deltay := trunc(largeurtypo * nbpointmm);
            figentite^.x := margegauchetypo;
            valeurentite^.pointsuivant := figentite;
            valeurentite := figentite;
            pointentite^.typesorte := textefigure;
          end;
        if pointdocument^.tyfeuille <> chapitre then
          begin
            if arbfor then
              begin
                mementite := pointentite;
                arbfor := false;
              end
              else
              valeurentite^.pointsuivant := pointentite;
          end;
      end;
    TRTARBREFORMATAGE(mementite,pointentite,arbfor,colonnage,larpoipa,
                      pointdocument^.pointsuccess1);
    TRTARBREFORMATAGE(mementite,pointentite,arbfor,colonnage,larpoipa,
                      pointdocument^.pointsuccess2);
    TRTARBREFORMATAGE(mementite,pointentite,arbfor,colonnage,larpoipa,
                      pointdocument^.pointsuivant);
  end
end;
{*****************************************************************************}
```

```
procedure BREAKPAGE(var adrpremierepage : ptypographique;pdocument : pgeneral);
H

label 1,2;
var pointeurpage,mempointeurpage : ptypographique;
    nbpage : integer;
begin
  nbpage := 1;
  pointeurpage := adrpremierepage;
  mempointeurpage := adrpremierepage;
  while nbpage <= pdocument^.premiere do
    begin
      mempointeurpage := pointeurpage;
      pointeurpage := pointeurpage^.pointsuivant;
      nbpage := nbpage + 1;
    end;
  adrpremierepage := mempointeurpage;
  if pdocument^.premiere = pdocument^.derniere then
    begin
      adrpremierepage^.pointsuivant := nil;
      goto 1;
    end;
  while nbpage <= pdocument^.derniere do
    begin
      mempointeurpage := pointeurpage;
      pointeurpage := pointeurpage^.pointsuivant;
      if pointeurpage = nil then goto 2;
      nbpage := nbpage + 1;
    end;
  2 :mempointeurpage^.pointsuivant := nil;
  1 :end;

{*****************************************************************************}


procedure UCASSURE(var ptentite : ptypographique;mementite,entitecourante :
                   ptypographique;longueurnote,longueurdisponible,ecartmoyen,
                   mpmargesuperieure,nbentitepage,longpoipa:integer);

var memoireligne,memligne,memoirefigure,ligne,boutli : ptypographique;
    test,val1,nbent,mpecartoptimal,distance,decr,cligne : integer;

begin
  memoirefigure := nil;
  val1 := 0;
  decr := 0;
  cligne := 0;
  test := 0;
  nbent := 0;
  mpecartoptimal := 0;
  distance := 0;
  while entitecourante <> nil do
    begin
      if longueurdisponible  > entitecourante^.deltay + ecartmoyen then
        begin
          nbent := nbent + 1;
          entitecourante^.y := longueurdisponible - entitecourante^.deltay +
                               longueurnote;
          longueurdisponible := longueurdisponible - entitecourante^.deltay -
                                ecartmoyen;
          memoirefigure := entitecourante;
          entitecourante := entitecourante^.pointsuivant;
        end
        else
        begin
          if entitecourante^.typesorte <> figure then
            begin
              ligne := entitecourante^.pointsuccess;
              if nbentitepage = 1 then longueurdisponible := longueurdisponible

                                                             - ecartmoyen;
              while test >= 0 do
                begin
                  val1 := val1 + ligne^.deltay +
                          entitecourante^.interligneentite;
                  test := longueurdisponible - val1;
                  cligne := cligne + 1;

                  memoireligne := ligne;
                  ligne := ligne^.pointsuivant;
                end;
              entitecourante^.y := longueurnote;
              entitecourante^.deltay := val1 - memoireligne^.deltay -
                                        entitecourante^.interligneentite;
              ligne := entitecourante^.pointsuccess;
              while cligne - 1 > 0 do
                begin
                  memligne := ligne;
                  cligne := cligne - 1;
                  ligne := ligne^.pointsuivant;
                end;
              if memligne <> nil
                then memligne^.pointsuivant := nil
                else memoireligne := ligne;
              test := 0;
              new(ptentite);
              ptentite^.pointsuccess := memoireligne;
              ptentite^.pointsuivant := entitecourante^.pointsuivant;
              ptentite^.interligneentite := entitecourante^.interligneentite;
              while memoireligne <> nil do
                begin
                  test := test + memoireligne^.deltay +
                          ptentite^.interligneentite;
                  memoireligne := memoireligne^.pointsuivant;
                end;
```

```
                ptentite^.deltay := test - entitecourante^.interligneentite;
                ligne := entitecourante^.pointsuccess;
                decr := test;
                while ligne <> nil do
                  begin
                    ligne^.y := ligne^.y - decr;
                    boutli := ligne^.pointsuccess;
                    while boutli <> nil do
                      begin
                        boutli^.y := boutli^.y - decr;
                        boutli := boutli^.pointsuivant;
                      end;
                    ligne := ligne^.pointsuivant;
                  end;
                ptentite^.x := entitecourante^.x;
                ptentite^.typesorte := entitecourante^.typesorte;
                ptentite^.typo := entitetypo;
                entitecourante^.pointsuivant := nil;
                entitecourante := nil;
              end
              else
              begin
                ptentite := entitecourante;
                memoirefigure^.pointsuivant := nil;

                (* rearangement des autres entites *)

                distance := memoirefigure^.y - longueurnote;
                mpecartoptimal := ecartmoyen + trunc(distance / (nbent - 1));
                longueurdisponible := longpoipa - mpmargesuperieure -
                                      longueurnote;
                entitecourante := mementite;
                while entitecourante <> nil do
                  begin
                    entitecourante^.y := longueurdisponible -
                                         entitecourante^.deltay + longueurnote;
                    longueurdisponible := longueurdisponible -
                                          entitecourante^.deltay -
                                          mpecartoptimal;
                    entitecourante := entitecourante^.pointsuivant;
                  end;
              end;
    end;
  end;
end;

{*****************************************************************************}

procedure UPLACEMENTNOTE(restenote,ecartminimum : integer;
                         var adrpremierenote : ptypographique;
                         pdocument : pgeneral);

var pentite,pligne,pboutli,intermediaire,indicenote : ptypographique;
    positionx,positiony,largmot,i,largeur,hauteur : integer;
    cara : char;
    sorte : tyfor;
begin
  sorte := boutlignetypo;
  CREERRECORD(pboutli,11,normal,rien,sorte,1500);
  sorte := lignetypo;
  CREERRECORD(pligne,11,normal,rien,sorte,1500);
  sorte := entitetypo;
  CREERRECORD(pentite,11,normal,rien,sorte,1500);
  pentite^.pointsuccess := pligne;
  pligne^.pointsuccess := pboutli;
  pboutli^.stringtypographique[0] := chr(23);
  i := 1;
  while i < 24 do
    begin
      pboutli^.stringtypographique[i] := '-';
      i := i + 1;
    end;
  cara := '-';
  EXAMEN(hauteur,largeur,cara,normal,11);
  largmot := largeur * 23;
  positionx := round((pdocument^.largeurfeuille * nbpointmm) / 2) -
               round(largmot / 2) - round(rognage * nbpointmm);
  positiony := restenote - round(1.5 * ecartminimum);
  GERERRECORD(0,0,largmot,hauteur,23,0,0,pboutli,boutlignetypo);
  GERERRECORD(0,0,largmot,hauteur,1,0,0,pligne,lignetypo);
  GERERRECORD(positionx,positiony,largmot,hauteur,1,0,0,pentite,
              entitetypo);
  intermediaire := adrpremierenote;
  adrpremierenote := pentite;
  pentite^.pointsuivant := intermediaire;
  indicenote := intermediaire;
  indicenote^.y := restenote - indicenote^.deltay - (3 * ecartminimum);
  restenote := restenote - indicenote^.deltay - (3 * ecartminimum);
  indicenote := indicenote^.pointsuivant;
  while indicenote <> nil do
  begin
    indicenote^.y := restenote - indicenote^.deltay - ecartminimum;
    restenote := restenote - indicenote^.deltay - ecartminimum;
    indicenote := indicenote^.pointsuivant;
  end;
end;

{*****************************************************************************}
```

```
procedure URESERVATIONNOTE(var adrpremierenote,entitecourante,
                           memoireentite,mpadrnote,mementite : ptypographique;
                           var ierenote : boolean;
                           var longueurnote,longueurdisponible : integer;
                           ecartminimum : integer);

label 1,2;
begin
  if ierenote then
    begin
      adrpremierenote := entitecourante;
      while entitecourante^.typesorte = notebaspage do
        begin
          mpadrnote := entitecourante;
          longueurnote := longueurnote + entitecourante^.deltay +
                          ( 3 *  ecartminimum);
          ngueurdisponible := longueurdisponible - entitecourante^.deltay -
                              (ecartminimum * 3);
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then goto 1;
        end;
      if memoireentite <> nil
        then  memoireentite^.pointsuivant := entitecourante;
        drnote^.pointsuivant := nil;
       erenote := false;
    end
    else
    begin
      mpadrnote^.pointsuivant := entitecourante;
      while entitecourante^.typesorte = notebaspage do
        begin
          mpadrnote := entitecourante;
          longueurnote := longueurnote + entitecourante^.deltay + ecartminimum;
          longueurdisponible := longueurdisponible - entitecourante^.deltay -
                                ecartminimum;
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then goto 2;
        end;
      2:if memoireentite <> nil then memoireentite^.pointsuivant :=
                                     entitecourante;
      mpadrnote^.pointsuivant := nil;
    end;
end;
(*****************************************************************************)

procedure UPOSITIONY1(entitecourante,mementite : ptypographique;
                      longueurdisponible,longpoipa,ecartmoyen,
                      mpmargesuperieure,longueurnote : integer);

begin
  entitecourante := mementite;
  if entitecourante^.typesorte = notebaspage then entitecourante :=
                                  entitecourante^.pointsuivant;
  longueurdisponible := longpoipa - mpmargesuperieure - longueurnote;
  while (entitecourante <> nil) do
    begin
      entitecourante^.y := longueurdisponible - entitecourante^.deltay +
                           longueurnote;
      longueurdisponible := longueurdisponible - ecartmoyen -
                            entitecourante^.deltay;
      entitecourante := entitecourante^.pointsuivant;
    end;
end;
(*****************************************************************************)

procedure UPOSITIONY2(var entitecourante : ptypographique;
                      var distance,mpcompteur : integer;
                      mementite : ptypographique;
                      longueurdisponible,longpoipa,mpmargesuperieure,
                      mpintermediaire,nbentitepage,longueurnote : integer);

var mpderniere : ptypographique;

begin
  mpcompteur := 0;
  entitecourante := mementite;
  longueurdisponible := longpoipa - mpmargesuperieure - longueurnote;
  while nbentitepage >0 do
    begin
      mpderniere := entitecourante;
      entitecourante^.y := longueurdisponible - entitecourante^.deltay +
                           longueurnote;
      distance := entitecourante^.y;
      longueurdisponible := longueurdisponible - entitecourante^.deltay -
                            mpintermediaire;
      entitecourante := entitecourante^.pointsuivant;
      mpcompteur := mpcompteur + 1;
      nbentitepage := nbentitepage - 1;
    end;
  mpderniere^.pointsuivant := nil;
end;
(*****************************************************************************)
```

```
procedure UPOSITIONY3(var entitecourante : ptypographique;
                      var distance,mpcompteur : integer;
                      mementite : ptypographique;
                      longueurdisponible,longpoipa,mpmargesuperieure,
                      mpintermediaire,nbentitepage,longueurnote : integer);

var mpderniere : ptypographique;
begin
  mpcompteur := Ø;
  entitecourante := mementite;
  longueurdisponible := longpoipa - mpmargesuperieure  - longueurnote;
  while nbentitepage - 1 > Ø do
    begin
      mpderniere := entitecourante;
      entitecourante^.y := longueurdisponible - entitecourante^.deltay +
                           longueurnote;
      distance := entitecourante^.y;
      longueurdisponible := longueurdisponible - entitecourante^.deltay -
                            mpintermediaire;
      entitecourante := entitecourante^.pointsuivant;
      mpcompteur := mpcompteur + 1;
      nbentitepage := nbentitepage - 1;
    end;
  mpderniere^.pointsuivant := nil;
end;

(*************************************************************************)

procedure UCREERPAGE(var adrpremierepage,adrdernierepagecree,ptpage :
                     ptypographique ;
                     var ierepage : boolean;
                     adrpremierenote,mementite : ptypographique);

var mpmem,mpmemoire : ptypographique;

begin
  new(ptpage);
  if ierepage then
    begin
      adrpremierepage := ptpage;
      adrdernierepagecree := ptpage;
      ierepage := false;
    end
    else
    begin
      adrdernierepagecree^.pointsuivant := ptpage;
      adrdernierepagecree := ptpage;
    end;
  ptpage^.typo := pagetypo;
  ptpage^.x := Ø;
  ptpage^.y := Ø;
  ptpage^.pointsuivant := nil;
  ptpage^.pointsuccess := mementite;
  mpmemoire := mementite;
  while mpmemoire <> nil do
    begin
      mpmem := mpmemoire;
      mpmemoire := mpmemoire^.pointsuivant;
    end;

   mpmem^.pointsuivant := adrpremierenote;
end;

 *************************************************************************)

procedure SEULESURPAGE(var adrpremierepage,adrdernierepagecree,mementite :
                       ptypographique;
                       pdocument : pgeneral;
                       var ierepage : boolean);

var entitecourante,entite,res,adrpremierenote,memoireentite,adrierfigure,
    mpadrnote,ptpage,prec : ptypographique;
    test,ierenote : boolean;
    debut,longueurdisponible,longueurnote,ecartminimum,mpmargesuperieure,
    longueur, nbentitepage,ecartmoyen,mpcompteurnote,longpoipa,
    margesup: integer;

begin
  ecartmoyen          := round (pdocument^.interentdocu * nbpointmm);
  ecartminimum        := ecartmoyen - trunc(Ø.5 * ecartmoyen);
  test                := true;
  ierenote            := true;
  ierepage            := true;
  mpcompteurnote      := Ø;
  longueur            := Ø;
```

```
  debut                 := 0;
  nbentitepage          := 0;
  longueurnote          := trunc(pdocument^.margeinferieure * nbpointmm);
  margesup              := trunc(pdocument^.margesuperieure * nbpointmm);
  longpoipa             := round(pdocument^.longueurfeuille * nbpointmm);
  adrpremierenote       := nil;
  adrpremierepage       := nil;
  entite                := nil;
  entitecourante        := mementite;
  while entitecourante^.typesorte <> entete  do
    begin
      if entitecourante^.typesorte = notebaspage then
        begin
          URESERVATIONNOTE(adrpremierenote,entitecourante,memoireentite,
                           mpadrnote,mementite,ierenote,longueurnote,
                           longueurdisponible,ecartminimum);
        end
        else
        begin
          longueur := longueur + entitecourante^.deltay;
          memoireentite := entitecourante;
          entitecourante := entitecourante^.pointsuivant;
          nbentitepage := nbentitepage + 1
        end;
    end;
  res := memoireentite^.pointsuivant;
  memoireentite^.pointsuivant := nil;
  longueurdisponible := trunc(((longpoipa - margesup - longueurnote) / 2);
  longueur := longueur + trunc(((nbentitepage - 1) * ecartmoyen) / 2);
  debut := longueurdisponible + longueur;
  entite := mementite;

  (* procedure differente de placement des y etant donne le centrage
     sur la page; il faut d'abord placer la premiere *)

  entite^.y := debut;
  debut := debut - entite^.deltay - ecartmoyen;

  (* debut := debut - ecartmoyen si on veut le meme ecart entre titre et
  le reste sinon l'ecart est celui du titre entre celui-ci et le reste *)

  entite := entite^.pointsuivant;
  while (nbentitepage - 1)  > 0 do
    begin
      nbentitepage := nbentitepage - 1;
      entite^.y := debut - entite^.deltay;

      debut := debut - entite^.deltay - ecartmoyen;
      entite := entite^.pointsuivant;
    end;
  if adrpremierenote <> nil then
  UPLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,pdocument);
  UCREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,
             adrpremierenote,mementite);
  mementite := res;
end;

(*************************************************************************)


procedure UNECOLO(var adrpremierepage : ptypographique;adrdernierepagecree,
                  mementite : ptypographique;pdocument :pgeneral;
                  ierepage,seulesurp : boolean);

var ecartminimum,ecartmaximum,mpmargesuperieure,mpmargeinferieure,
    longueurdisponible,mpcompteur,nbentitepage,mpintermediaire,longueurnote,
    longpoipa,larpoipa,ecartmoyen,mpresteentite,distance,ecartoptimal,
    restenote : integer;
    adrpremierenote,memoireentite,mpadrnote,entitecourante,memory,ptentite,
    ptpage,mpprochain,memsaut : ptypographique;
    test,ierenote,saut :boolean;

(* initialisation des variables *)

begin
  ecartmoyen          := round (pdocument^.interentdocu * nbpointmm);
  ecartminimum        := ecartmoyen - trunc(0.5 * ecartmoyen);
  ecartmaximum        := 2 * ecartmoyen;
  mpmargesuperieure   := round(pdocument^.margesuperieure * nbpointmm);
  mpmargeinferieure   := round(pdocument^.margeinferieure * nbpointmm);
  longpoipa           := round(pdocument^.longueurfeuille * nbpointmm);
  larpoipa            := round(pdocument^.largeurfeuille * nbpointmm);

(* fin des initialisations des variables invariantes a plusieurs pages *)

while mementite <> nil do
  begin
```

```
  (* reinitialisations des variables locales de traitement de mise en page *)

    ierenote              := true;
    test                  := true;
    saut                  := false;
    mpcompteur            := Ø;
    ecartoptimal          := Ø;
    nbentitepage          := Ø;
    mpintermediaire       := Ø;
    longueurdisponible    := longpoipa - mpmargesuperieure - mpmargeinferieure;
    mpresteentite         := Ø;
    longueurnote          := mpmargeinferieure;
    restenote             := longueurnote;
    memory                := nil;
    adrpremierenote       := nil;
    mpprochain            := nil;
    memoireentite         := nil;

(* debut de la procedure: on parcourt toutes les entites et  arret *)
(* quand on a fini ou qu'on a plus de place sur la page*)

  entitecourante := mementite;
  while ((longueurdisponible >Ø )  and (test)) do
    begin
      if entitecourante <> nil then
        begin
          if  (entitecourante^.sautpagetypographique) and (memoireentite
                              <> nil) then
            begin
              saut := true;
              memsaut := entitecourante;

              entitecourante := nil;
              test := false;
              memoireentite^.pointsuivant := nil;
            end
            else
            begin
              if entitecourante^.typesorte = notebaspage then
                begin
                  URESERVATIONNOTE(adrpremierenote,entitecourante,
                                   memoireentite,mpadrnote,mementite,ierenote,
                                     longueurnote,longueurdisponible,
                                   ecartminimum);
                end
                else
                begin
                  longueurdisponible := longueurdisponible -
                                        entitecourante^.deltay - ecartmoyen;
                  memoireentite := entitecourante;
                  entitecourante := entitecourante^.pointsuivant;
                  nbentitepage := nbentitepage + 1
                end;
            end;
        end
        else test := false
  end;

(* traitement des differents cas possibles *)

if ((entitecourante = nil) and (longueurdisponible >Ø)) then
  begin

    (* cas ou la page n'est pas toute remplie *)

    if adrpremierenote <> nil
      then UPLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,pdocument);

    UPOSITIONY1(entitecourante,mementite,longueurdisponible,longpoipa,
                ecartmoyen,mpmargesuperieure,longueurnote);
    UCREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,
               adrpremierenote,mementite);
    if saut then
      begin
        memsaut^.sautpagetypographique := false;
        mementite := memsaut
      end
      else mementite := nil;
  end
  else
  begin
    if longueurdisponible <= Ø  then
      begin
        (* cas de chevauchement d'une entite des lors calcul pour voir
           si l'on peut mettre l'entite chevauchante sur la page en
           resserant les autres *)

        mpintermediaire := (longueurdisponible + (nbentitepage *
                           (ecartmoyen - ecartminimum)));

          (* on a dans mpintermediaire une valeur,
             si >Ø alors en resserant un peu tout mettre sur une page
             sinon on doit casser l'entite c-a-d la mettre toute sur
             la page suivante *)

        if( mpintermediaire >=Ø) and (nbentitepage >1) then
          begin
            mpintermediaire := ecartminimum + trunc(mpintermediaire
                               / nbentitepage );
            UPOSITIONY2(entitecourante,distance,mpcompteur,mementite,
                        longueurdisponible,longpoipa,mpmargesuperieure,
                        mpintermediaire,nbentitepage,longueurnote);
            mpprochain := entitecourante;
            ecartoptimal := trunc(( distance - longueurnote ) /
```

```
                                (mpcompteur - 1));
          mpintermediaire := mpintermediaire + ecartoptimal;
          nbentitepage := mpcompteur;
          UPOSITIONY2(entitecourante,distance,mpcompteur,mementite,
                      longueurdisponible,longpoipa,mpmargesuperieure,
                      mpintermediaire,nbentitepage,longueurnote);
          if adrpremierenote <> nil then
            UPLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,
                           pdocument);
          UCREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,
                     adrpremierenote,mementite);
          mementite := mpprochain;
        end
        else
        begin

          (*cas ou il n'y a pas assez de place,il reste a calculer
            si l'entite chevauchante doit etre placee sur la page
            suivante ou si l'on peut placer une partie sur la page
            et placer le reste sur l'autre page.
            principe:on calcul la place que prend les entites(chevauchantes
            non comprises) avec l'ecart maximum : si la plasce disponible
            resultante de ce calcul est <Ø on ne peut en placer une
            partie,sinon on la casse *)

          entitecourante := mementite;
          longueurdisponible := longpoipa -  mpmargesuperieure - longueurnote;


          mpresteentite := nbentitepage;
          while(nbentitepage - 1 > Ø) do
            begin
              nbentitepage := nbentitepage - 1;
              longueurdisponible := longueurdisponible -
                                    entitecourante^.deltay - ecartmaximum ;
              entitecourante := entitecourante^.pointsuivant;
            end;
          if ((longueurdisponible < Ø) and (mpresteentite > 1)) then
            begin

              (* cas ou l'entite chevauchante ne peut etre placee sur la
                 page courante,donc placement sur la page suivante et
                 rangement des entites precedentes *)

              nbentitepage := mpresteentite;
              mpintermediaire := (longueurdisponible + (nbentitepage *
                                 (ecartmaximum - ecartmoyen)));
              mpintermediaire := ecartmoyen;
              UPOSITIONY3(entitecourante,distance,mpcompteur,mementite,
                          longueurdisponible,longpoipa,mpmargesuperieure,
                          mpintermediaire,nbentitepage,longueurnote);
              ecartoptimal := trunc ((distance - longueurnote) /
                                     (mpcompteur - 1));
              mpintermediaire := mpintermediaire + ecartoptimal;
              nbentitepage := mpresteentite;
              mpprochain := entitecourante;
              UPOSITIONY3(entitecourante,distance,mpcompteur,mementite,
                          longueurdisponible,longpoipa,mpmargesuperieure,
                          mpintermediaire,nbentitepage,longueurnote);
              if adrpremierenote <> nil then
                UPLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,
                               pdocument);
              UCREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,
                         ierepage,adrpremierenote,mementite);
              mementite := mpprochain;
            end
            else
            begin
              nbentitepage := mpresteentite;

              (* cas ou l'on effectue la cassure de la derniere entite *)

              if adrpremierenote <> nil then
                begin
                  restenote := longueurnote;
                  UPLACEMENTNOTE(restenote,ecartminimum,adrpremierenote,
                                 pdocument);
                end;
              longueurdisponible := longpoipa - mpmargesuperieure -
                                    longueurnote;
              entitecourante := mementite;
              UCASSURE(ptentite,mementite,entitecourante,longueurnote,
                       longueurdisponible,ecartmoyen,mpmargesuperieure,
                       nbentitepage,longpoipa);
              UCREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,
                         ierepage,adrpremierenote,mementite);
              mementite := ptentite;
            end
        end
      end
    end
  end
end;

(*****************************************************************************)
```

```
procedure SAUTPAGE(var memoireentite,ptpageentite : ptypographique;
                   longueurdisponible,longueurnote,longueurfigure : integer);

var pligne,pmemoire,memligne,boutli : ptypographique;
    somme,test,decr,cligne : integer;

begin
  decr         := 0;
  cligne       := 0;
  test         := test;
  somme        := 0;
  pligne       := memoireentite^.pointsuccess;
  while test >=0 do
    begin
      somme := somme + pligne^.deltay + memoireentite^.interligneentite;
      test := longueurdisponible - somme;
      cligne := cligne + 1;
      pmemoire := pligne;
      pligne := pligne^.pointsuivant;
    end;
  memoireentite^.y := longueurnote + longueurfigure;
  memoireentite^.deltay := somme - pmemoire^.deltay -
                           memoireentite^.interligneentite;
  memoireentite^.x := 945;
  pligne := memoireentite^.pointsuccess;
  while cligne - 1 > 0 do
    begin
      memligne := pligne;
      cligne := cligne - 1;
      pligne := pligne^.pointsuivant;
    end;
  if memligne <> nil
    then memligne^.pointsuivant := nil
    else pmemoire := pligne;
  new(ptpageentite);
  ptpageentite^.typo := entitetypo;
  ptpageentite^.pointsuccess := pmemoire;
  ptpageentite^.pointsuivant := memoireentite^.pointsuivant;
  ptpageentite^.interligneentite := memoireentite^.interligneentite;
  ptpageentite^.typesorte := memoireentite^.typesorte;
  test := 0;
  while pmemoire <> nil do
    begin
      test := test + pmemoire^.deltay + memoireentite^.interligneentite;
      pmemoire := pmemoire^.pointsuivant;
    end;
  ptpageentite^.deltay := test - memoireentite^.interligneentite;
  pligne := memoireentite^.pointsuccess;

  decr := test;
  while pligne <> nil do
    begin
      pligne^.y := pligne^.y - decr;
      boutli := pligne^.pointsuccess;
      while boutli <> nil do
        begin
          boutli^.y := boutli^.y - decr;
          boutli := boutli^.pointsuivant;
        end;
      pligne := pligne^.pointsuivant;
    end;
  memoireentite^.pointsuivant := nil;
end;

{*****************************************************************************}
```

```
procedure COLONNAGE(var entitecourante : ptypographique;mementite,memoryentite
                    : ptypographique;
                    longueurnote,longueurdisponible,ecartmoyen,mpcompteur,
                    longpoipa,longueurfigure,mpmargesuperieure : integer);

label 1;
var memoireligne,ptentite,memligne,ligne,mpenti,memmpenti,
    boutli : ptypographique;
    test,vali,decr,cligne,taille: integer;
    fanion : boolean;

begin
  entitecourante := mementite;
  memligne       := nil;
  memmpenti      := nil;
  vali           := vali;
  decr           := 0;
  cligne         := 0;
  test           := 0;
  fanion         := false;
  while mpcompteur - 1 > 0  do
    begin
      memmpenti := entitecourante;
      entitecourante := entitecourante^.pointsuivant;
      mpcompteur := mpcompteur - 1;
    end;
  if memmpenti <> nil then
    begin
      longueurdisponible := memmpenti^.y - longueurnote - longueurfigure -
                            ecartmoyen;
      ligne := entitecourante^.pointsuccess;
      taille := ligne^.deltay + entitecourante^.interligneentite;
      if longueurdisponible < 0 then
        begin
          fanion := true;
          goto 1;
        end;
    end
    else
    begin
      longueurdisponible := longpoipa - mpmargesuperieure - longueurnote -
                            longueurfigure;
      entitecourante := memoryentite;
    end;
  ligne := entitecourante^.pointsuccess;
  while (test >= 0 )  do
    begin
      vali := vali + ligne^.deltay + entitecourante^.interligneentite;
      test := longueurdisponible - vali;
      memoireligne := ligne;
      cligne := cligne +1;
      ligne := ligne^.pointsuivant;
    end;
  entitecourante^.y := longueurnote + longueurfigure;
  entitecourante^.deltay := vali  - memoireligne^.deltay -
                            entitecourante^.interligneentite;
  ligne := entitecourante^.pointsuccess;
  while cligne - 1 > 0 do
    begin
      memligne := ligne;
      cligne := cligne - 1;
      ligne := ligne^.pointsuivant;
    end;
  if memligne <> nil
    then memligne^.pointsuivant := nil
    else memoireligne := ligne;
  test := 0;
  new(ptentite);
  ptentite^.pointsuccess := memoireligne;
  ptentite^.pointsuivant := entitecourante^.pointsuivant;
  ptentite^.interligneentite := entitecourante^.interligneentite;
  while memoireligne <> nil do
    begin
      test := test + memoireligne^.deltay + entitecourante^.interligneentite;
      memoireligne := memoireligne^.pointsuivant;
    end;
  ptentite^.deltay := test - entitecourante^.interligneentite;
  decr := test;
  ligne := entitecourante^.pointsuccess;
  while ligne <> nil do
    begin
      ligne^.y := ligne^.y - decr;
      boutli := ligne^.pointsuccess;
      while boutli <> nil do
        begin
          boutli^.y := boutli^.y - decr;
          boutli := boutli^.pointsuivant;
        end;
      ligne := ligne^.pointsuivant;
    end;
  ptentite^.typo := entitetypo;
  ptentite^.typesorte := entitecourante^.typesorte;
  entitecourante^.pointsuivant := ptentite;
  1 :if fanion then entitecourante := memoryentite else
  entitecourante := ptentite;
end;

(*****************************:*************************************************)
```

```
procedure PLACEMENTNOTE(var longueurnote,ecartminimum : integer;
                        var adrpremierenote : ptypographique; pdocument :
                        pgeneral);

var pentite,pligne,pboutli,intermediaire,indicenote : ptypographique;
    positionx,positiony,largmot,i,largeur,hauteur : integer;
    cara : char;
    sorte : tyfor;

begin
  sorte := boutlignetypo;
  CREERRECORD(pboutli,11,normal,rien,sorte,1500);
  sorte := lignetypo;
  CREERRECORD(pligne,11,normal,rien,sorte,1500);
  sorte := entitetypo;
  CREERRECORD(pentite,11,normal,rien,sorte,1500);
  pentite^.pointsuccess := pligne;
  pligne^.pointsuccess := pboutli;
  pboutli^.stringtypographique[0] := chr(23);
  i := 1;
  while i < 24 do
    begin
      pboutli^.stringtypographique[i] := '-';
      i := i + 1;
    end;
  cara := '-';
  EXAMEN(hauteur,largeur,cara,normal,11);
  largmot := largeur * 23;
  positionx := round((pdocument^.largeurfeuille * nbpointmm) / 2) -
               round(largmot / 2) - round(rognage * nbpointmm);
  positiony := longueurnote - round(1.5 * ecartminimum);
  GERERRECORD(0,0,largmot,hauteur,23,0,0,pboutli,
              boutlignetypo);
  GERERRECORD(0,0,largmot,hauteur,1,0,0,pligne,
              lignetypo);
  GERERRECORD(positionx,positiony,largmot,hauteur,1,0,0,pentite,
              entitetypo);
  intermediaire := adrpremierenote;
  adrpremierenote := pentite;
  pentite^.pointsuivant := intermediaire;
  indicenote := intermediaire;
  indicenote^.y := longueurnote - indicenote^.deltay - ( 3 * ecartminimum);
  longueurnote := longueurnote - indicenote^.deltay - (3 * ecartminimum);
  indicenote := indicenote^.pointsuivant;
  while indicenote <> nil do
    begin
      indicenote^.x := 0;
      indicenote^.y := longueurnote - indicenote^.deltay - ecartminimum;
      longueurnote := longueurnote - indicenote^.deltay - ecartminimum;
      indicenote := indicenote^.pointsuivant;
    end;
end;

{*****************************************************************************}

procedure PLACEMENTFIGURE(var adrierfigure : ptypographique;
                          longueurfigure,ecartminimum,longueurnote : integer);

var indicefigure : ptypographique;

begin
  indicefigure := adrierfigure;
  while indicefigure <> nil do
    begin
      indicefigure^.y := ((longueurfigure - indicefigure^.deltay) -
                          ecartminimum) + longueurnote;
      longueurfigure := longueurfigure - indicefigure^.deltay - ecartminimum;
      indicefigure := indicefigure^.pointsuivant;
    end;
end;

{*****************************************************************************}

procedure RESERVATIONNOTE(var adrpremierenote,entitecourante,memoireentite,
                          mpadrnote : ptypographique;
                          var ierenote : boolean;
                          var longueurnote,longueurdisponible : integer;
                          ecartminimum : integer);
label 1,2;

begin
  if ierenote then
    begin
      adrpremierenote := entitecourante;
      while entitecourante^.typesorte = notebaspage do
        begin
          mpadrnote := entitecourante;
          longueurnote := longueurnote + entitecourante^.deltay +
                          (3 * ecartminimum);
          longueurdisponible := longueurdisponible - (2 * entitecourante^.
                                deltay) - (6 * ecartminimum);
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then goto 1;
        end;
      1: memoireentite^.pointsuivant := entitecourante;
```

```
      mpadrnote^.pointsuivant := nil;
      ferenote := false;
    end
    else
    begin
      mpadrnote^.pointsuivant := entitecourante;
      while entitecourante^.typesorte = notebaspage do
        begin
          mpadrnote := entitecourante;
          longueurnote := longueurnote + entitecourante^.deltay + ecartminimum;

          longueurdisponible := longueurdisponible - (2 * entitecourante^.
                                deltay) - (2 * ecartminimum);
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then goto 2;
        end;
      2: memoireentite^.pointsuivant := entitecourante;
      mpadrnote^.pointsuivant := nil;
    end;
end;
(*************************************************************************)

procedure RESERVATIONFIGURE(var adrierfigure,entitecourante,mpadrfig,
                            memoireentite : ptypographique;
                            var longueurdisponible,longueurfigure : integer;
                            var mpierfig : boolean;
                            ecartminimum : integer);

label 1,2;

begin
  if mpierfig then
    begin
      adrierfigure := entitecourante;
      while ((entitecourante^.typesorte = figure) or
             (entitecourante^.typesorte = textefigure)) do
        begin
          mpadrfig := entitecourante;
          longueurfigure := longueurfigure + entitecourante^.deltay +
                            ecartminimum;
          longueurdisponible := longueurdisponible - (2 * (entitecourante^.
                                deltay + ecartminimum));
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then goto 1;
        end;
      1: if memoireentite <> nil then memoireentite^.pointsuivant :=
                                      entitecourante;
      mpadrfig^.pointsuivant := nil;
      mpierfig := false;
    end
    else
    begin
      mpadrfig^.pointsuivant := entitecourante;
      while ((entitecourante^.typesorte = figure) or
             (entitecourante^.typesorte = textefigure)) do
        begin
          mpadrfig := entitecourante;
          longueurfigure := longueurfigure + entitecourante^.deltay +
                            ecartminimum;
          longueurdisponible := longueurdisponible - (2 *(entitecourante^.
                                deltay + ecartminimum));
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then goto 2;
        end;
      2: if memoireentite <> nil then memoireentite^.pointsuivant :=
                                      entitecourante;
      mpadrfig^.pointsuivant := nil;
    end;
end;

(*************************************************************************)
```

```
procedure POSITIONY1(var memoryentite : ptypographique; entitecourante,
                     mementite,ptentite : ptypographique;
                     longueurdisponible,longpoipa,ecartmoyen,mpmargesuperieure,

                     mpmargeinferieure,ecartminimum,longueurnote,ecartmaximum,
                     longueurfigure,somme: integer);

var position,mpcompteur : integer;
    test : boolean;

begin
  mpcompteur := 0;
  test := true;
  position := 0;
  entitecourante := mementite;
  longueurdisponible := longpoipa - mpmargesuperieure -longueurnote -
                        longueurfigure;
  while (entitecourante <> nil) do
    begin
      while  (longueurdisponible >0 ) and (test)  do
        begin
          entitecourante^.y := longueurdisponible - entitecourante^.deltay +
                               longueurnote + longueurfigure;
          entitecourante^.x := position;
          memoryentite := entitecourante;
          longueurdisponible := longueurdisponible - ecartmoyen -
                                entitecourante^.deltay;
          entitecourante := entitecourante^.pointsuivant;
          if entitecourante = nil then test := false;
          mpcompteur := mpcompteur + 1;
        end;
    if longueurdisponible < 0 then
      begin
        COLONNAGE(entitecourante,mementite,memoryentite,longueurnote,
                  longueurdisponible,ecartmoyen,mpcompteur,longpoipa,
                  longueurfigure,mpmargesuperieure);
        longueurdisponible := longpoipa - mpmargesuperieure -
                              longueurnote - longueurfigure - somme;
        position := 945;
        test := true;
      end;
  end;
end;

(*****************************************************************************)

procedure CREERPAGE(var adrpremierepage,adrdernierepagecree,ptpage :
                    ptypographique;
                    var ierepage : boolean;
                    adrpremierenote,adrierfigure,mementite : ptypographique);

var mpmem,mpmemoire,mpfig,mpmemfig : ptypographique;

begin
  new(ptpage);
  if ierepage then
    begin
      adrpremierepage := ptpage;
      adrdernierepagecree := ptpage;
      ierepage := false;
    end
    else
    begin
      adrdernierepagecree^.pointsuivant := ptpage;
      adrdernierepagecree := ptpage;
    end;
  ptpage^.typo := pagetypo;
  ptpage^.x := 0;
  ptpage^.y := 0;
  ptpage^.pointsuivant := nil;
  ptpage^.pointsuccess := mementite;
  mpmemoire := mementite;
  while mpmemoire <> nil do
    begin
      mpmem := mpmemoire;
      mpmemoire := mpmemoire^.pointsuivant;
    end;

  (* lien des notes *);

  if adrpremierenote <> nil then
    begin
      mpmem^.pointsuivant := adrpremierenote;

     (* lien des figures *);

     mpfig := adrpremierenote;
     while mpfig <> nil do
       begin
         mpmemfig := mpfig;
         mpfig := mpfig^.pointsuivant;
       end;
     mpmemfig^.pointsuivant := adrierfigure;
    end
    else
    mpmem^.pointsuivant := adrierfigure;
end;

(*****************************************************************************)
```

```
procedure DEUXCOLO(var adrpremierepage,adrdernierepagecree : ptypographique;
                       mementite : ptypographique;pdocument :pgeneral;
                       ierepage,seulesurp : boolean);


var ecartminimum,somme,ecartmaximum,mpmargesuperieure,mpmargeinferieure,
    longueurdisponible,nbentitepage,mpintermediaire,longueurnote,
    longueurfigure,ecartmoyen,longpoipa,larpoipa,mpresteentite,ert,
    distance,ecartoptimal,restenote,mpcompteurinter: integer;
    adrpremierenote,memoireentite,mpadrnote,entitecourante,memoryentite,
    ptentite,ptpage,mpprochain,mpadrfig,mpentite,memsaut,adrierfigure,
    mpderniere,ptpageentite,uneentite : ptypographique;
    test,ierenote,mpierfig,saut :boolean;

(* initialisation des variables pour toutes les pages *)

begin
  ecartmoyen              := round(pdocument^.interentdocu * nbpointmm);
  ecartminimum            := ecartmoyen - trunc(0.5 * ecartmoyen);
  ecartmaximum            := 2 * ecartmoyen;
  mpmargesuperieure       := round(pdocument^.margesuperieure * nbpointmm);
  mpmargeinferieure       := round(pdocument^.margeinferieure * nbpointmm);
  longpoipa               := round(pdocument^.longueurfeuille * nbpointmm);
  larpoipa                := round(pdocument^.largeurfeuille * nbpointmm);

  (* initialisation des variables pour le traitemenr d'une page *)

  while mementite <> nil do
  begin
    ierenote              := true;
    mpierfig              := true;
    test                  := true;
    saut                  := false;
    somme                 := 0;
    ecartoptimal          := 0;
    mpcompteurinter       := 0;
    nbentitepage          := 0;
    mpresteentite         := 0;
    mpintermediaire       := 0;
    longueurfigure        := 0;
    ert                   := ecartmoyen;
    restenote             := mpmargeinferieure;
    longueurnote          := mpmargeinferieure;
    longueurdisponible    := 2 * (longpoipa - mpmargesuperieure -
                              mpmargeinferieure);
    adrpremierenote       := nil;
    adrierfigure          := nil;
    uneentite             := nil;
    mpderniere            := nil;
    ptentite              := nil;
    memoryentite          := nil;
    mpprochain            := nil;
    ptpageentite          := nil;
    memoireentite         := nil;

    (* fin des initialisation *)

    entitecourante := mementite;
    while ((longueurdisponible >0 ) and (test)) do
      begin
        if entitecourante <> nil then
          begin
            if (entitecourante^.sautpagetypographique) and
              (memoireentite <> nil)  then
            begin
              saut := true;
              memsaut := entitecourante;
              entitecourante := nil;
              memoireentite^.pointsuivant := nil;
            end
            else
            begin
              if entitecourante^.typesorte = notebaspage then
                begin
                  RESERVATIONNOTE(adrpremierenote,entitecourante,
                                  memoireentite,mpadrnote,ierenote,
                                  longueurnote,longueurdisponible,
                                  ecartminimum);
                end
                else
                begin
                  if ((entitecourante^.typesorte = figure) or
                  (entitecourante^.typesorte = textefigure)) then
                    begin
                      RESERVATIONFIGURE(adrierfigure,entitecourante,mpadrfig,

                                        memoireentite,longueurdisponible,
                                        longueurfigure,mpierfig,
                                        ecartminimum);
                    end
```

```
                    else
                    begin
                      if longueurdisponible < ecartmaximum
                        then ert := ecartminimum;
                      if (entitecourante^.typesorte = titre) or
                         (entitecourante^.typesorte = date) or
                         (entitecourante^.typesorte = auteur) or
                         (entitecourante^.typesorte = abstract) then
                        begin
                          somme := somme + entitecourante^.deltay +
                                   ecartmoyen;
                          longueurdisponible := longueurdisponible -
                                                (2 *(entitecourante^.deltay +

                                                 ecartmoyen));
                          nbentitepage := nbentitepage + 1;
                          memoireentite := entitecourante;
                          entitecourante := entitecourante^.pointsuivant;
                        end
                        else

                        begin
                          longueurdisponible := longueurdisponible -
                                                entitecourante^.deltay -
                                                ecartmoyen;
                          memoireentite := entitecourante;
                          entitecourante := entitecourante^.pointsuivant;
                          nbentitepage := nbentitepage + 1
                        end;
                    end;
                  end;
                end;
              end
        else
        test := false
end;
if ((entitecourante = nil) and (longueurdisponible >Ø)) then
begin
  POSITIONY1(memoryentite,entitecourante,mementite,ptentite,
             longueurdisponible,longpoipa,ecartmoyen,mpmargesuperieure,
             mpmargeinferieure,ecartminimum,longueurnote,ecartmaximum,
             longueurfigure,somme);
  if adrierfigure <> nil then
    PLACEMENTFIGURE(adrierfigure,longueurfigure,ecartminimum,longueurnote);
  if adrpremierenote <> nil then
    PLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,pdocument);
  CREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,
             adrpremierenote,adrierfigure,mementite);
  if saut then
    begin
      memsaut^.sautpagetypographique := false;
      mementite := memsaut
    end
    else  mementite := nil;
  end
  else
  begin
   if nbentitepage = 1 then
     begin
       longueurdisponible := longpoipa - longueurnote - mpmargesuperieure -
                             longueurfigure;
       SAUTPAGE(memoireentite,ptpageentite,longueurdisponible,longueurnote,
                longueurfigure);
       memoireentite^.pointsuivant := ptpageentite;
       memoireentite^.x := Ø;
       longueurdisponible := longpoipa - mpmargesuperieure - longueurnote -
                             longueurfigure;
       memoireentite := ptpageentite;
       SAUTPAGE(memoireentite,ptpageentite,longueurdisponible,longueurnote,
                longueurfigure);
       if adrierfigure <> nil then
         PLACEMENTFIGURE(adrierfigure,longueurfigure,ecartminimum,
                         longueurnote);
       if adrpremierenote <> nil then
         PLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,pdocument);
         CREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,
                   adrpremierenote,adrierfigure,mementite);
       mementite := ptpageentite;
     end
     else
     begin
       mpentite := mementite;
       mpcompteurinter := nbentitepage;
       while mpcompteurinter - 1 >Ø do
         begin
           mpderniere := mpentite;
           mpentite := mpentite^.pointsuivant;
           mpcompteurinter := mpcompteurinter - 1;
         end;
       mpderniere^.pointsuivant := nil;
       POSITIONY1(memoryentite,entitecourante,mementite,ptentite,
                  longueurdisponible,longpoipa,ecartmoyen,mpmargesuperieure,
                    mpmargeinferieure,ecartminimum,longueurnote,ecartmaximum,
```

```
                longueurfigure,somme);
        memoryentite^.pointsuivant := memoireentite;
        if memoryentite^.x <> Ø then
          begin
            longueurdisponible := memoryentite^.y - longueurnote -
                                  longueurfigure - ecartmoyen;
            if (longueurdisponible - ecartmaximum) > Ø then
              begin
                SAUTPAGE(memoireentite,ptpageentite,longueurdisponible,
                         longueurnote,longueurfigure);
                if adrierfigure <> nil then
                PLACEMENTFIGURE(adrierfigure,longueurfigure,ecartminimum,
                                longueurnote);
                if adrpremierenote <> nil then
                PLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,
                              pdocument);
                CREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,

                          adrpremierenote,adrierfigure,mementite);
                mementite := ptpageentite;
              end
               else
               begin
                 if adrierfigure <> nil then
                   PLACEMENTFIGURE(adrierfigure,longueurfigure,ecartminimum,
                                   longueurnote);
                 if adrpremierenote <> nil then
                   PLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,
                                 pdocument);
                 CREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,
                           ierepage,adrpremierenote,adrierfigure,mementite);
                 memoryentite^.pointsuivant := nil;
                 mementite := memoireentite;
               end;
          end
          else
          begin
            longueurdisponible := memoryentite^.y - longueurfigure -
                                  longueurnote - ecartmoyen;
            SAUTPAGE(memoireentite,ptpageentite,longueurdisponible,
                     longueurnote,longueurfigure);
            memoireentite^.x := Ø;
            memoireentite^.pointsuivant := ptpageentite;
            longueurdisponible := longpoipa - longueurfigure -
                                  mpmargesuperieure - longueurnote - somme;
            memoireentite := ptpageentite;
            SAUTPAGE(memoireentite,ptpageentite,longueurdisponible,
                     longueurnote,longueurfigure);
            if adrierfigure <> nil then
              PLACEMENTFIGURE(adrierfigure,longueurfigure,ecartminimum,
                              longueurnote);
            if adrpremierenote <> nil then
              PLACEMENTNOTE(longueurnote,ecartminimum,adrpremierenote,
                            pdocument);
            CREERPAGE(adrpremierepage,adrdernierepagecree,ptpage,ierepage,
                      adrpremierenote,adrierfigure,mementite);
            mementite := ptpageentite;
          end;
      end;
    end;
  end;
end;

(*****************************************************************************)

procedure MISEENPAGE(var adrpremierepage : ptypographique;mementite:
                     ptypographique; pdocument : pgeneral);

var adrdernierepagecree : ptypographique;
    ierepage,seulesurp : boolean;

begin
  adrdernierepagecree := nil;
  ierepage := true;
  seulesurp := true;
  if pdocument^.colonnage then
    begin
      if pdocument^.seule then
        begin
          SEULESURPAGE(adrpremierepage,adrdernierepagecree,mementite,
                       pdocument,ierepage);
          seulesurp := false;
        end;
      DEUXCOLO(adrpremierepage,adrdernierepagecree,mementite,pdocument,
               ierepage,seulesurp);
    end
    else
    begin
      if pdocument^.seule then
        begin
          SEULESURPAGE(adrpremierepage,adrdernierepagecree,mementite,
                       pdocument,ierepage);
          seulesurp := false;
        end;
      UNECOLO(adrpremierepage,adrdernierepagecree,mementite,
              pdocument,ierepage,seulesurp);
    end;
end;

(*****************************************************************************)
```

```
procedure NUMEROTATION(pdocument : pgeneral;var adrpremierepage :
                       ptypographique);

var pagecourante,dernieereentite,pentitecourante,pentite,pligne,pboutli :
    ptypographique;
    positionx,diviseur,nb,numpage,chiffrecara,largmot,i,largeur,memposition,
    hauteur : integer;
    cara : char;
    sorte : tyfor;

begin
  numpage := pdocument^.numpage;
  pagecourante := adrpremierepage;
  if pdocument^.seule = true then pagecourante := pagecourante^.pointsuivant;
  while pagecourante <> nil do
    begin
      largmot := Ø;
      nb := 3;
      numpage := numpage + 1;
      chiffrecara := numpage;
      sorte := boutlignetypo;
      CREERRECORD(pboutli,11,normal,rien,sorte,15ØØ);
      sorte := lignetypo;
      CREERRECORD(pligne,11,normal,rien,sorte,15ØØ);
      sorte := entitetypo;
      CREERRECORD(pentite,11,normal,rien,sorte,15ØØ);
      pentite^.pointsuccess := pligne;
      pligne^.pointsuccess := pboutli;
      pboutli^.stringtypographique[3] := '=';
      diviseur := trunc(chiffrecara / 1ØØ);
      if diviseur > Ø then
        begin
          pboutli^.stringtypographique[3] := chr(diviseur + 48);
          nb := nb + 1;
          chiffrecara := chiffrecara - (diviseur * 1ØØ);
        end;
      diviseur := trunc(chiffrecara / 1Ø);
      if diviseur > Ø then
        begin
          if pboutli^.stringtypographique[3] <> '='
            then pboutli^.stringtypographique[4] := chr(diviseur + 48)
            else pboutli^.stringtypographique[3] := chr(diviseur + 48);
          nb := nb + 1;
          chiffrecara := chiffrecara - (diviseur * 1Ø);
        end;
      pboutli^.stringtypographique[Ø] := chr(nb + 2);
      pboutli^.stringtypographique[1] := '-';
      pboutli^.stringtypographique[2] := ' ';
      pboutli^.stringtypographique[nb] := chr(chiffrecara + 48);
      pboutli^.stringtypographique[nb + 1] := ' ';
      pboutli^.stringtypographique[nb + 2] := '-';
      i := 3;
      while pboutli^.stringtypographique[i] <> ' ' do
        begin
          cara := pboutli^.stringtypographique[i];
          EXAMEN(hauteur,largeur,cara,normal,11);
          largmot := largmot + largeur;
          i := i + 1;
        end;
      positionx := round((pdocument^.largeurfeuille * nbpointmm) / 2) -
                   round(largmot / 2) - round(rognage * nbpointmm);
      memposition := nb + 2;
      GERERRECORD(Ø,Ø,largmot,hauteur,memposition,1Ø,2,pboutli,
                  boutlignetypo);
      GERERRECORD(Ø,Ø,largmot,hauteur,1,1Ø,2,pligne,
                  lignetypo);
      GERERRECORD(positionx,1ØØ,largmot,hauteur,1,Ø,Ø,pentite,
                  entitetypo);
      pentitecourante := pagecourante^.pointsuccess;
      if pdocument^.superieure then
        begin
          pagecourante^.pointsuccess := pentite;
          pentite^.pointsuivant := pentitecourante;
          pentite^.y := 269Ø;
        end
        else
        begin
          while pentitecourante <> nil do
            begin
              dernieereentite := pentitecourante;
              pentitecourante := pentitecourante^.pointsuivant;
            end;
          dernieereentite^.pointsuivant := pentite;
        end;
      pagecourante := pagecourante^.pointsuivant;
    end;
end;
(*****************************************************************************)
```

```
procedure  TABLEMATIERE(var fertab : ptypographique;
                        arbstruc : pgeneral;arbfor : ptypographique);

var cptent,mrs,i,val,largeurblanc,haut,larg,hauteurmot,largeurmot,
    longdisponible,valniv,nbniv,vali,intertabs,mrgtab,mrdtab,poltab,
    nbppage,longpoipa,larpoipa : integer;
    fotab : typefonte;
    soutab : typesoulignement;
    memppage,pentite,pligne,pboutli,ppage,vpboutli,lpboutli,
    mementite: ptypographique;
    sorte : tyfor;
    seule,ierligne,trouver : boolean;

{*****************************************************************************}

procedure GESTIONMATIERE(arbstruc : pgeneral);

var mot : pmot;
procedure PROCENT(var vali,nbppage : integer;arbfor : ptypographique;
                  cptent : integer);

var page,entite : ptypographique;

begin
  while (vali ) <> cptent  do
    begin
      if arbfor^.typo = pagetypo then
        begin
          page := arbfor;
          arbfor := arbfor^.pointsuccess;
          nbppage := nbppage + 1;
        end;
    if arbfor^.typo = entitetypo then
        begin
          if arbfor^.typesorte = entete then vali := vali + 1;
          if arbfor^.pointsuivant = nil
            then arbfor := page^.pointsuivant
            else arbfor := arbfor^.pointsuivant;
        end;
    end;
end;

begin
  if ((arbstruc^.tyfeuille = entete)
              and (arbstruc^.niveauchapitre <= valniv)) then

    begin
      ierligne := true;
      mementite := pentite;
      mot := arbstruc^.pointmot;
      while mot <> nil do
        begin
          mot^.soulignement := soutab;
          mot^.police := poltab;
          mot^.fonte := fotab;
          mot^.sautligne := false;
          mot := mot^.pointmot;
        end;
      TRAITEMENTCLASSIQUE(pentite,arbstruc^.pointmot,larpoipa,mrgtab,mrdtab,
                          poltab,trunc(intertabs / 4),0,ierligne,false,fotab,
                          soutab);
      pentite^.y := longdisponible - pentite^.deltay;
      longdisponible := longdisponible - pentite^.deltay - intertabs;
      if (longdisponible - mrs < 0 ) then
        begin
          memppage := ppage;
          sorte := pagetypo;
          CREERRECORD(ppage,poltab,fotab,soutab,sorte,0);

                (* faire le lien *)

          ppage^.pointsuccess := pentite;
          longdisponible := longpoipa - pentite^.deltay;
          pentite^.y := longdisponible;
          memppage^.pointsuivant := ppage;
        end
        else
        begin
          pentite^.y := longdisponible;
          mementite^.pointsuivant := pentite;
          mementite := pentite;
        end;

        (* creation d'un record avec le numero de ppage *)

      sorte := boutlignetypo;
      CREERRECORD(pboutli,poltab,fotab,soutab,sorte,0);
      pligne := pentite^.pointsuccess;

        (* recherche du dernier boutdeligne de l'entete *)
```

```
      while pligne <> nil do
        begin
          lpboutli := pligne^.pointsuccess;
          pligne := pligne^.pointsuivant;
        end;
      while lpboutli <> nil do
        begin
          vpboutli := lpboutli;
          lpboutli := lpboutli^.pointsuivant;
        end;
      vpboutli^.pointsuivant := pboutli;

        (* recherche du numero de ppage dans l'arbre de formatage *)

      nbppage := Ø;
      if  seule then nbppage := nbppage - 1 ;
      vali := Ø;
      PROCENT(vali,nbppage,arbfor,cptent);
      val := Ø;
      i := 1;
      if nbppage  >= 1ØØ then
        begin
          val := trunc(nbppage / 1ØØ);
          pboutli^.stringtypographique[i] := chr(val + 48);
          i := i + 1;
        end;
      if nbppage >= 1Ø then
        begin
          val := trunc(nbppage / 1Ø) - (val * 1Ø);
          pboutli^.stringtypographique[i] := chr(val + 48);
          i := i + 1;
        end;
      if i = 3
        then nbppage := nbppage - ( 1Ø * trunc(nbppage / 1Ø)) -
                          (1ØØ * trunc(nbppage/1ØØ))
        else nbppage := nbppage - ( 1Ø * trunc(nbppage / 1Ø));
      pboutli^.stringtypographique[i] := chr(nbppage + 48);
      pboutli^.nombreelement := i;
      GERERRECORD(mrgtab + larpoipa + 1Ø,Ø,Ø,Ø,i,Ø,Ø,pboutli,sorte);
    end;
end;
(*****************************************************************************)

procedure TRTTABLE(arbstruc : pgeneral);

begin
 if arbstruc <> nil then
   begin
     GESTIONMATIERE(arbstruc);
     if arbstruc^.tyfeuille = entete then cptent := cptent + 1;
     TRTTABLE(arbstruc^.pointsuccess1);
     TRTTABLE(arbstruc^.pointsuccess2);
     TRTTABLE(arbstruc^.pointsuivant);
   end;
end;
```

```
{*****************************************************************************}
{*                        PROCEDURE TABLEMATIERE - corps                     *}
{*****************************************************************************}
...
begin
  valniv := arbstruc^.nivtab;
  mrs := trunc(nbpointmm * arbstruc^.margeinferieure);
  intertabs := trunc(nbpointmm * arbstruc^.intertab);
  mrgtab := TRUNC(nbpointmm * arbstruc^.mrgmat);
  mrdtab := trunc(nbpointmm * arbstruc^.mrdmat);
  fotab := arbstruc^.fotab;
  soutab := arbstruc^.soutab;
  poltab := arbstruc^.poltab;
  nbniv := valniv;

  seule := arbstruc^.seule;
  longpoipa := trunc((arbstruc^.longueurfeuille * nbpointmm) -
                     (arbstruc^.margesuperieure * nbpointmm));
  longdisponible := longpoipa ;
  larpoipa := trunc(arbstruc^.largeurfeuille * nbpointmm) - mrgtab   - mrdtab ;

  sorte := pagetypo;
  CREERRECORD(ppage,poltab,fotab,soutab,sorte,0);

        (* creation du premier pboutli comprenant le titre *)

  iertab := ppage;
  sorte := entitetypo;
  CREERRECORD(pentite,poltab,fotab,soutab,sorte,0);
  mementite := pentite;
  sorte := lignetypo;
  CREERRECORD(pligne,poltab,fotab,continu,sorte,0);
  sorte := boutlignetypo;
  CREERRECORD(pboutli,poltab,fotab,continu,sorte,0);
  pboutli^.stringtypographique[1] := ' ';
  pboutli^.stringtypographique[2] := 'T';
  pboutli^.stringtypographique[3] := 'a';
  pboutli^.stringtypographique[4] := 'b';
  pboutli^.stringtypographique[5] := 'l';
  pboutli^.stringtypographique[6] := 'e';
  pboutli^.stringtypographique[7] := ' ';
  pboutli^.stringtypographique[8] := 'd';
  pboutli^.stringtypographique[9] := 'e';
  pboutli^.stringtypographique[10] := 's';
  pboutli^.stringtypographique[11] := ' ';
  pboutli^.stringtypographique[12] := 'm';
  pboutli^.stringtypographique[13] := 'a';
  pboutli^.stringtypographique[14] := 't';
  pboutli^.stringtypographique[15] := 'i';
  pboutli^.stringtypographique[16] := chr(19);
  pboutli^.stringtypographique[17] := 'r';
  pboutli^.stringtypographique[18] := 'e';
  pboutli^.stringtypographique[19] := 's';
  pboutli^.nombreelement := 19;
  nbppage := 0;
  largeurmot := 0;
  hauteurmot := 0;
  for i := 1 to 19 do
    begin
      EXAMEN(haut,larg,pboutli^.stringtypographique[i],fotab,poltab);
      if hauteurmot < haut then hauteurmot := haut;
      largeurmot := largeurmot + larg;
    end;
  largeurblanc := largeurmot div 19;
  GERERRECORD(0,0,largeurmot + (2 * largeurblanc),hauteurmot,17,largeurblanc,
              3,pboutli,sorte);
  longdisponible := longdisponible - hauteurmot;
  sorte := lignetypo;
  GERERRECORD(0,0,largeurmot + (2 * largeurblanc),hauteurmot,17,largeurblanc,
              3,pligne,sorte);
  sorte := entitetypo;
  GERERRECORD((larpoipa - largeurmot) div 2,longdisponible, largeurmot +
          (2 * largeurblanc),hauteurmot,17,largeurblanc,3,pentite,sorte);
  ppage^.pointsuccess := pentite;
  pentite^.pointsuccess := pligne;
  pligne^.pointsuccess := pboutli;
  longdisponible := longdisponible - 100;
  cptent := 1;
  TRTTABLE(arbstruc);
end;
```

```
(*****************************************************************************)
(*                        PROCEDURE ARBRE FORMATAGE - corps                  *)
(*****************************************************************************)



begin

  TABR1Ø;TABG11;TABI11;TABR11;TAB14;
WRITELN('JE TRAVAILLE . . .');
  arbfor := true;
  mementite := nil;
  pointentite := nil;
  TRTARBREFORMATAGE(mementite,pointentite,arbfor,pointdoc^.colonnage ,
        trunc(nbpointmm * pointdoc^.largeurfeuille),pointdoc^.pointsuivant);
WRITELN('JE TRAVAILLE . . .');
  MISEENPAGE(pointpage,mementite,pointdoc);
  if pointdoc^.pagination then NUMEROTATION(pointdoc,pointpage);
  if pointdoc^.tablematiere then
    begin
      TABLEMATIERE(iertab,pointdoc,pointpage);
      adrpage := pointpage;
      while adrpage <> nil do
        begin
          mempage := adrpage;
          adrpage := adrpage^.pointsuivant;
        end;
      mempage^.pointsuivant := iertab;
    end;
WRITELN('JE TRAVAILLE . . .');
end;
```

```
(*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)
(*                PROCEDURES DE CONSTRUCTION DU FORMAT FDD
(*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)

procedure FDDTRAITEMENT (var fertabrecord:ptabrecord;mempage:ptypographique);

type  ptableau = ^tableau;
      tableau  = record
        adlonglist : integer;
        addatalist : integer;
        adlongdata : integer;
        lglist : integer;
        pointeursuivant : ptableau;
      end;

var   adpage,zero,nbpage,i,cptpage,compteurtabrecord,compteurinf : integer;
      tabrecord : ptabrecord;
      tab,adrfirsta : ptableau;
      valpage : ptypographique;
      firsta : boolean;
      v11,v12 :byte;
      v111 : word;

(*****************************************************************************)


procedure MISEAMOINSUN (var miseamoinsuntravail : block);

var miseamoinsuni : integer;
    miseamoinsunbyte : byte;
    miseamoinsunword : word;

begin
  miseamoinsuni :=0;
  while miseamoinsuni <=7 do
    begin
      miseamoinsunbyte[miseamoinsuni] := 1;
      miseamoinsuni := miseamoinsuni + 1;
    end;
  miseamoinsuni := 0;
  while miseamoinsuni <=1 do
    begin
      miseamoinsunword[miseamoinsuni] := miseamoinsunbyte;
      miseamoinsuni := miseamoinsuni + 1;
    end;
  miseamoinsuni := 0;
  while miseamoinsuni <=255 do
    begin
      miseamoinsuntravail[miseamoinsuni] := miseamoinsunword;
      miseamoinsuni := miseamoinsuni + 1
    end;
end;
(*****************************************************************************)

procedure REMPLIRWORD (var titi1 : word;var1,var2:byte);

begin
  titi1[0] := var1;
  titi1[1] := var2;
end;
(*****************************************************************************)


procedure VALEUR(var titi2 : byte;vt : integer);

var i : integer;

begin
  i := 0;
  while i <= 7 do  begin titi2[i] := 0; i := i + 1 end;
  if vt <0 then
    begin
      i := 0;
      while i <=7 do begin titi2[i] := 1; i := i + 1 end
    end;
  if vt - 128 >= 0 then begin vt := vt - 128; titi2[0] := 1 end;
  if vt - 64 >= 0  then begin vt := vt - 64;  titi2[1] := 1 end;
  if vt - 32 >= 0  then begin vt := vt - 32;  titi2[2] := 1 end;
  if vt - 16 >= 0  then begin vt := vt - 16;  titi2[3] := 1 end;
  if vt - 8 >= 0   then begin vt := vt - 8;   titi2[4] := 1 end;
  if vt - 4 >= 0   then begin vt := vt - 4;   titi2[5] := 1 end;
  if vt - 2 >= 0   then begin vt := vt - 2;   titi2[6] := 1 end;
  if vt - 1 >= 0   then begin vt := vt - 1;   titi2[7] := 1 end
end;
(*****************************************************************************)
```

```
procedure TRANSFORMEWORD (var titi31,titi32 :byte;vt : integer);

var j,i : integer;

begin
  j := Ø;
  while j <=7 do begin titi31[j] := Ø; titi32[j] := Ø; j := j + 1 end ;
  if vt < Ø then
    begin
      j := Ø;
      while j <=15 do begin titi32[j] := 1; titi31[j] := 1;j := j + 1 end
    end;
  if vt - 32768 >= Ø then begin vt := vt - 32768; titi31[Ø] := 1 end;
  if vt - 16384 >= Ø then begin vt := vt - 16384; titi31[1] := 1 end;
  if vt - 8192 >= Ø  then begin vt := vt - 8192;  titi31[2] := 1 end;
  if vt - 4Ø96 >= Ø  then begin vt := vt - 4Ø96;  titi31[3] := 1 end;
  if vt - 2Ø48 >= Ø  then begin vt := vt - 2Ø48;  titi31[4] := 1 end;
  if vt - 1Ø24 >= Ø  then begin vt := vt - 1Ø24;  titi31[5] := 1 end;
  if vt - 512 >= Ø   then begin vt := vt - 512;   titi31[6] := 1 end;
  if vt - 256 >= Ø   then begin vt := vt - 256;   titi31[7] := 1 end;
  if vt - 128 >= Ø   then begin vt := vt - 128;   titi32[Ø] := 1 end;
  if vt - 64 >= Ø    then begin vt := vt - 64;    titi32[1] := 1 end;
  if vt - 32 >= Ø    then begin vt := vt - 32;    titi32[2] := 1 end;
  if vt - 16 >= Ø    then begin vt := vt - 16;    titi32[3] := 1 end;
  if vt - 8 >= Ø     then begin vt := vt - 8;     titi32[4] := 1 end;
  if vt - 4 >= Ø     then begin vt := vt - 4;     titi32[5] := 1 end;
  if vt - 2 >= Ø     then begin vt := vt - 2;     titi32[6] := 1 end;
  if vt - 1 >= Ø     then begin vt := vt - 1;     titi32[7] := 1 end;
end;
(*****************************************************************************)

function INCR(incrcompteur : integer):integer;

var  rec : ptabrecord;
     i,j,k,vart : integer;
     wor : word;
     byt : byte;

begin
  if compteurinf <> Ø then
    begin
      if ((compteurinf + 1) mod 256) = Ø then
        begin
          rec := tabrecord;
          new(tabrecord);
          rec^.pointsuivant := tabrecord;
          compteurtabrecord := compteurtabrecord + 1;
          MISEAMOINSUN(tabrecord^.zonetravail);
          tabrecord^.pointsuivant := nil;
          tabrecord^.numero := compteurtabrecord ;
        end;
    end;
  INCR :=incrcompteur + 1;
end;
(*****************************************************************************)


procedure MISERECORD(x, y : integer);

var  v121,v122 : byte;
     v11 : word;
     adresse : integer;

begin
  VALEUR(v121,x);
  VALEUR(v122,y);
  REMPLIRWORD(v11,v121,v122);
  adresse := compteurinf mod 256;
  tabrecord^.zonetravail[adresse] := v11;
  compteurinf := INCR(compteurinf);
end;
(*****************************************************************************)


procedure CONSTRDIRECTORY ( nbpage : integer);

var  var1,var2 : byte;
     i,x,y,z : integer;
     v113 : word;

        procedure recherchedate(var x,y,z : integer);

        begin
          x := 12;
          y := 12;
          z := 1982;
        end;

begin
  new(tabrecord);
  MISEAMOINSUN(tabrecord^.zonetravail);
```

```
  compteurtabrecord := compteurtabrecord + 1;
  iertabrecord := tabrecord;
  tabrecord^.pointsuivant :=nil;
  tabrecord^.numero := compteurtabrecord;
        (* mot numero Ø : type de document    *)
  if nbpage = 1 then MISERECORD(Ø,Ø)  else MISERECORD(-1,-1);
        (* mot numero 1 : nombre de tabrecord    *)
  compteurinf := INCR(compteurinf);
        (* mot numero 2 : nombre de part    *)
  if nbpage = 1 then MISERECORD(Ø,Ø) else MISERECORD(Ø,Ø);
        (* mot numero 3 : debut part directory    *)
  compteurinf := INCR(compteurinf);
        (* mot numero 4 : nombre de tabrecord    *)
  if nbpage = 1 then MISERECORD(Ø,1);
  RECHERCHEDATE(X,Y,Z);
        (* mot numero 5 : date    *)
  MISERECORD(Ø,x);
        (* mot numero 6 : date    *)
  MISERECORD(Ø,y);
        (* mot numero 7 : date    *)
  TRANSFORMEWORD(var1,var2,z);
  REMPLIRWORD(v113,var1,var2);
  tabrecord^.zonetravail[compteurinf mod 256] := v113;
  compteurinf := INCR(compteurinf);
        (* mot numero 8 : -1    *)
  MISERECORD(-1,-1);
        (* mot numero 9 : -1    *)
  MISERECORD(-1,-1);
        (* mot numero 1Ø : -1    *)
  MISERECORD(-1,-1);
        (* mot numero 11 :  -1   *)
  MISERECORD(-1,-1);
        (* mot numero 12 :  -1   *)
  MISERECORD(-1,-1);
        (* mot numero 13 :  -1   *)
  MISERECORD(-1,-1);
        (* mot numero 14 : final printer   *)
  MISERECORD(Ø,Ø);
        (* mot numero 15 : type de feuille    *)
  MISERECORD(Ø,3);
  for i := 16 to 63 do compteurinf := INCR(compteurinf);
  if nbpage <> 1 then compteurinf := INCR(compteurinf);
end;
(*****************************************************************************)


procedure CONSTRUCPARTDIRECTORY;

var i : integer;
    rec : ptabrecord;

begin
  MISERECORD(Ø,Ø);
  MISERECORD(Ø,Ø);
  MISERECORD(Ø,Ø);
  MISERECORD(Ø,Ø);
end;

(*****************************************************************************)


procedure TRTPAGEDATA(var longdata : integer;tab : ptableau;var ierpas,ier :
                      boolean; elementcourant : ptypographique;var var1 :char);


var     tr,valinterim : integer;
        var2 : char;
        val1 : ptabrecord;
        v111 : word;
        v11,v12: byte;


begin
  if elementcourant <> nil then
    begin
      if elementcourant^.typo = boutlignetypo then
        begin
          tr := Ø;
          while tr < elementcourant^.nombreelement do
            begin
              tr := tr + 1;
              if ier then
                begin
                  var1 := elementcourant^.stringtypographique[tr];
                  ier := false;
                end
                  else
                begin
                  var2 := elementcourant^.stringtypographique[tr];
                  ier := true;
                  MISERECORD(ord(var1),ord(var2));
                end;
            end;
        end;
```

```
        if elementcourant^.typo = entitetypo then
          begin
                (* mise de l'adresse du data list au bon endroit *)
            valinterim := compteurinf;
            while (valinterim - 256) >= Ø do valinterim := valinterim - 256;
            val1 := fertabrecord;
            while (tab^.addatalist - 256)  >= Ø do
              begin
                val1 := val1^.pointsuivant;
                tab^.addatalist := tab^.addatalist - 256;
              end;
            TRANSFORMEWORD(v11,v12,(compteurinf - adpage) * 2 );
            REMPLIRWORD(v111,v11,v12);
            val1^.zonetravail[tab^.addatalist] := v111;
            tab := tab^.pointeursuivant;
          end;
        TRTPAGEDATA(longdata,tab,ferpas,fer,elementcourant^.pointsuccess,var1);

        if (( elementcourant^.typo = lignetypo ) or (elementcourant^.typo =
           boutlignetypo) )then
          begin
            if not fer then
              begin
                MISERECORD(ord(VAR1),32);
                longdata := longdata - 1;
              end ;
            fer := true;
          end;
          TRTPAGEDATA(longdata,tab,ferpas,fer,elementcourant^.pointsuivant,var1);


    end;
end;
(*****************************************************************************)


procedure TRTLIGNE (var nbcarentite: integer;memligne : ptypographique);

var boutli : ptypographique;
    tysouli,tys,ty,fon,val,fon1,varincr : integer;
    v11 : word;
    v121,v122: byte;

begin
  if memligne <> nil then
    begin
      boutli := memligne^.pointsuccess;
      varincr := memligne^.largeurvoulue - memligne^.deltax;
      while boutli <> nil do
        begin
          nbcarentite := nbcarentite + boutli^.nombreelement;

          (* realisation du <VARIABLE SPACED CHAR >*)

          MISERECORD(231,Ø);

          (*  realisation du <FONT>*)

          case boutli^.policetypographique of
             1Ø : fon := 1;
             11 : begin
                    case boutli^.fontetypographique of
                      gras : fon := 32;
                      italique : fon := 16;
                      normal : fon := Ø;
                    end;
                    fon := fon + 2;
                  end;
             14 : fon := 3;
             otherwise fon := 2;
          end;
          if boutli^.policetypographique = 1Ø then fon1 := Ø else fon1 := 1;
          MISERECORD(112 + fon1,fon);

          (* realisation du <SET INC WORDSPACE> *)

          if (boutli^.nombreblanc - varincr) >= Ø then
            begin
              MISERECORD(244,varincr);
              varincr := Ø;
            end
            else
            begin
              MISERECORD(244,boutli^.nombreblanc);
              varincr := varincr - boutli^.nombreblanc;
            end;
          MISERECORD(Ø,memligne^.VALUESPACE);

          (* realisation du <SET X > *)

          val := Ø;
          while (boutli^.x - 256) > Ø do
            begin
              val := val + 1;
              boutli^.x := boutli^.x - 256;
            end;
          MISERECORD(val,BOUTLI^.X);
```

```
          (* realisation du <SET Y>*)

          val := Ø;
          while (boutli^.y - 256) > Ø do
            begin
              val := val + 1;
              boutli^.y := boutli^.y - 256;
            end;
          MISERECORD(32 + val,BOUTLI^.Y);

          (* realisation du <HIGHTLIGNTHING > *)

          if boutli^.soulignementtypographique <> rien then
            begin
              if boutli^.soulignementtypographique = continu then ty := Ø;
              if boutli^.soulignementtypographique = discontinu then ty := 16;
              tys:= 2 ;
              if (boutli^.fontetypographique = gras)  then tys := tys + 1;
              if (boutli^.policetypographique = 14)  then tys := 4;
              if boutli^.soulignementtypographique = superieure then
                begin
                  ty := 48;
                  tys := 2;
                end;
              tysouli := 128 + ty + tys;
              MISERECORD(229,tysouli);
            end;

          (* realisation du <SHOW CAR> *)

          if (boutli^.nombreelement /2 )= (boutli^.nombreelement div 2)
            then MISERECORD(24Ø,BOUTLI^.NOMBREELEMENT)
            else MISERECORD(24Ø,boutli^.nombreelement +1);

          (* realisation de la fin du <HIGHTLIGHTING> *)

          if boutli^.soulignementtypographique <> rien then
            MISERECORD(229,tysouli -128);
          boutli := boutli^.pointsuivant;
        end;
      TRTLIGNE(nbcarentite,memligne^.pointsuivant);
    end;
  end;

(*****************************************************************************)


procedure  TRTPAGELIST( mempointeur: ptypographique);

var  memtab : ptableau;
     fon,i,j,valinterim,nbcarentite: integer;
     var1 ,var2 : byte;
     v : byte;vl13 :word;
     val1 : ptabrecord;

begin
  if mempointeur <> nil then
    begin
      if mempointeur^.typo = entitetypo then
        begin

              (* creation d'une nouvelle ligne dans le tableau d'adresse *)

          memtab := tab;
          new(tab);
          if firsta then
            begin
              adrfirsta := tab ;
              firsta := false
            end
            else memtab^.pointeursuivant := tab;
          tab^.lglist := compteurinf;
          tab^.pointeursuivant := nil;
          tab^.adlonglist := compteurinf;
                          (* premier mot *)
          compteurinf := INCR(compteurinf);
                          (* deuxieme mot : creation type,fonte info *)
          MISERECORD(Ø,Ø);
                          (* troisieme mot dl-type *)
          MISERECORD(Ø,Ø);
                          (* quatrieme,cinquieme mot : adresse dl-region *)
          MISERECORD(Ø,Ø);

          tab^.addatalist := compteurinf;
          compteurinf := INCR(compteurinf);
                          (* sixieme,septieme mot : longeur dl-region *)
          MISERECORD (Ø,Ø);
          tab^.adlongdata := compteurinf ;
          compteurinf := INCR(compteurinf);
                          (* huitieme mot : xe *)
          MISERECORD(Ø,Ø);
                          (* neuvieme mot : ye *)
          MISERECORD(Ø,Ø);
                          (* dixieme mot : xbb *)
          TRANSFORMEWORD(var1,var2,mempointeur^.x);
          REMPLIRWORD(vl13,var1,var2);
```

```
        tabrecord^.zonetravail[compteurinf mod 256] := v113;
        compteurinf := INCR(compteurinf);
                    (* onzieme mot : ybb *)
        TRANSFORMEWORD(var1,var2,mempointeur^.y);
        REMPLIRWORD(v113,var1,var2);
        tabrecord^.zonetravail[compteurinf mod 256] := v113;
        compteurinf := INCR(compteurinf);
                    (*douzieme mot : deltaxbb*)
        TRANSFORMEWORD(var1,var2,mempointeur^.deltax);
        REMPLIRWORD(v113,var1,var2);
        tabrecord^.zonetravail[compteurinf mod 256] := v113;
        compteurinf := INCR(compteurinf);
                    (*treizieme mot : deltaybb*)
        TRANSFORMEWORD(var1,var2,mempointeur^.deltay);
        REMPLIRWORD(v113,var1,var2);
        tabrecord^.zonetravail[compteurinf mod 256] := v113;
        compteurinf := INCR(compteurinf);
        nbcarentite := 0;
        TRTLIGNE (nbcarentite,mempointeur^.pointsuccess);
        MISERECORD(-1,-1);(* mot de separation du TRAILER et des commandes *)

        if mempointeur^.typo = entitetypo then
          begin
              (* mise dans le tableau de la longeur du data list = nombre
                          de caractere *)
            valinterim := compteurinf;
            while (valinterim - 256) >= 0 do valinterim := valinterim - 256;
            val1 := tertabrecord;
            while (tab^.adlongdata - 256) >= 0 do
              begin
                val1 := val1^.pointsuivant;
                tab^.adlongdata := tab^.adlongdata - 256;
              end;
            TRANSFORMEWORD(var1,var2,nbcarentite);
            REMPLIRWORD(v113,var1,var2);
            val1^.zonetravail[tab^.adlongdata] := v113;
          end;
        if mempointeur^.typo = entitetypo then
          begin
              (* mise dans l'entite list de la longeur de cette entite list *)


            valinterim := compteurinf;
            while (valinterim - 256) >= 0 do valinterim := valinterim - 256;
            val1 := tertabrecord;
            while (tab^.adlonglist - 256) >= 0 do
              begin
                val1 := val1^.pointsuivant;
                tab^.adlonglist := tab^.adlonglist - 256;
              end;
            TRANSFORMEWORD(var1,var2,(compteurinf - tab^.lglist) * 2);
            REMPLIRWORD(v113,var1,var2);
            val1^.zonetravail[tab^.adlonglist] := v113;
          end;
        TRTPAGELIST (mempointeur^.pointsuivant);
      end;
  end;
end;
{*****************************************************************************}

procedure PARCOURS(var cptpage : integer;mempage : ptypographique;
                       nbpage : integer);
var  rec : ptabrecord;
     i,adrdebutentitylist,adrierpart,cpt,longdata,j : integer;
     v11,v12 : byte;v111 : word;
     var1 : char;
     ierpas,ier : boolean;
     adrtableau : ptableau;

begin
  firsta := true;
  if mempage <> nil then
    begin
      cptpage := cptpage + 1;
      adpage := compteurinf;
      rec := tertabrecord;
      adrierpart := 60;
      rec := tertabrecord;

        (* inscription dans les parts directory de l'adresse du debut des
        informations relatives a cette page *)

      if nbpage > 1 then
        begin
          j := 1 + adrierpart + (cptpage * 4);
          while j >= 256 do
            begin
              rec := rec^.pointsuivant;
              j := j - 256;
            end;
          cpt := 1;
          TRANSFORMEWORD(v11,v12,compteurtabrecord - 1);
          REMPLIRWORD(v111,v11,v12);
          rec^.zonetravail[j] := v111;
        end;

        (* mise a jour du compteur d'informations *)

      adrdebutentitylist := compteurtabrecord;
      TRTPAGELIST(mempage^.pointsuccess);

       (* au cas ou une seule page on doit inscrire la longueur  de l'entity
                  list padding *)
```

```
      if nbpage = 1 then
        begin
          TRANSFORMEWORD(v11,v12,(compteurinf - adrdebutentitylist ) * 2);
          REMPLIRWORD(v111,v11,v12);
          iertabrecord^.zonetravail[13] := v111;
        end;
      longdata := Ø;
      ierpas := true;
      ier := true;
      MISERECORD(Ø,Ø);
      TRTPAGEDATA(longdata,adrfirsta,ierpas,ier,mempage^.pointsuccess,var1);
      for i := (compteurinf mod 256) to 255 do compteurinf :=
                INCR(compteurinf);
      if nbpage > 1 then
        begin
          TRANSFORMEWORD(v11,v12,compteurtabrecord - adrdebutentitylist);
          REMPLIRWORD(v111,v11,v12);
          rec^.zonetravail[j + 1] := v111;
        end;
      PARCOURS(cptpage,mempage^.pointsuivant,nbpage);
    end;
end;



(*****************************************************************************)
(*                        PROCEDURE FDDTRAITEMENT - corps                    *)
(*****************************************************************************)

begin
  valpage := mempage;
  nbpage := Ø;
  while valpage <> nil do
    begin
      nbpage := nbpage + 1;
      valpage := valpage^.pointsuivant;
    end;
  tabrecord := nil;
  compteurtabrecord :=Ø;
  compteurinf := Ø;
  adrfirsta := nil;
  firsta := true;
  CONSTRDIRECTORY(nbpage);
  if nbpage <> 1 then
    begin
      for i := 1 to nbpage do CONSTRUCPARTDIRECTORY;
      for i := compteurinf to 255 do compteurinf := INCR(compteurinf);
    end;
  cptpage := Ø;
  PARCOURS(cptpage,mempage,nbpage);
  TRANSFORMEWORD(v11,v12,compteurtabrecord);
  REMPLIRWORD(v111,v11,v12);
  iertabrecord^.zonetravail[1] := v111;
  if nbpage <> 1 then
    begin
      TRANSFORMEWORD(v11,v12,nbpage);
      REMPLIRWORD(v111,v11,v12);
      iertabrecord^.zonetravail[2] := v111;
    end;
end;
```

```
(*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)
(*                PROCEDURE D'ENCODAGE EN BINAIRE DU FORMAT FDD                  *)
(*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*)

procedure IMPRBIT(ptprre : ptabrecord);

var  compteur,impi1,impi2,impvariable : integer;
     impinterim : char;
     imp : ptabrecord;
     impzeroun : zeroun;
     impbyte : byte;
     impword : word;

(*****************************************************************************)

procedure TRANSFORME (var v1 : char; v2 : zeroun);

begin
if v2 = 1 then v1 := '1' else v1 := 'Ø';
end;

(*****************************************************************************)

begin
  compteur := Ø;
  rewrite(defini);
  imp := ptprre;
  while imp <> nil do
    begin
      impvariable := Ø ;
      while impvariable < 256 do
        begin
          impword := imp^.zonetravail[impvariable];
          impi1 := Ø;
          compteur := compteur + 1;
          while impi1 <= 1 do
            begin
              impi2 := Ø;
              impbyte := impword[impi1];
              while impi2 <= 7 do
                begin
                  impzeroun := impbyte [ impi2];
                  TRANSFORME(impinterim,impzeroun);

                  write(defini,impinterim);
                  impi2 := impi2 + 1;
                end;
              impi1 := impi1 + 1;
            end;
          writeln(defini);
          impvariable := impvariable + 1;
        end;
      imp := imp^.pointsuivant;
    end;
end;
```

```
(*****************************************************************************)
(*                        FIN DE DECLARATIONS DE PROCEDURE                   *)
(*****************************************************************************)

begin
test:= true;
                    (* indicateur de saut de page *)

csautaut := false;      csautdat := false;      csauten1 := false;
csauten2 := false;      csauten3 := false;      csauten4 := false;
csautfig := false;      csautlis := false;      csautnot := false;
csautpar := false;      csautres := false;      csauttit := false;
                        csauttab := false;

    (*  indicateur de centrage des elements de structure *)

ccentaut := true;       ccentdat := true;       ccenten1 := true;
ccenten2 := false;      ccenten3 := false;      ccenten4 := false;
ccentfig := true;       ccentlis := false;      ccentnot := false;
ccentpar := false;      ccentres := false;      ccenttit := true;
                        ccenttab := false;

                    (* indicateur booleen divers *)

ccolodoc := false;      cimpredu := false;      cindexdo := false;
cpagidoc := false;      cpagisup := false;      cseulpag := false;
                        ctabdoc  := false;

                    (* valeurs entieres diverses *)

cdecpara := 1Ø;         cdecalis := 1Ø;         cinferie := Ø;
cdernier := Ø;          cinteren := 5;          clargeur := 17Ø;
clongueur:= 27Ø;        cnumfig  := 1;          cnumnot  := 1;
cnumpag  := Ø;          cnument  := Ø;          ctabniv  := 2;

    (* valeur entiere de l'interligne des elements de structure *)

cinliaut := 3;          cinlidat := 4;          cinlien1 := 3;
cinlien2 := 3;          cinlien3 := 3;          cinlien4 := 2;
cinlifig := 2;          cinlilis := 3;          cinlinot := 1;
cinlipar := 2;          cinlires := 3;          cinlitit := 5;
                        cinlitab := 3;

    (* valeur entiere de la marge gauche des elements de structure *)

cmargaut := 2Ø;         cmargdat := 2Ø;         cmargen1 := 3Ø;
cmargen2 := 1Ø;         cmargen3 := 1Ø;         cmargen4 := 1Ø;
cmarglis := 1Ø;         cmargnot := 1Ø;         cmargpar := 1Ø;
cmargres := 2Ø;         cmargtit := 2Ø;         cmargtab := 2Ø;
                        cmargfig := 1Ø;

    (* valeur entiere de la marge droite des elements de structure *)

cmardaut := rognage + 1Ø;       cmarddat := rognage + 1Ø;
cmarden1 := rognage + 2Ø;       cmarden2 := rognage + Ø;
cmarden3 := rognage + Ø;        cmarden4 := rognage + Ø;
cmardfig := rognage + Ø;        cmardlis := rognage + Ø;
cmardnot := rognage + Ø;        cmardpar := rognage + Ø;
cmardres := rognage + 1Ø;       cmardtit := rognage + 1Ø;
            cmardtab := rognage + 1Ø;

    (* valeur entiere de la marge superieure / inferieure *)

cmardsud := 1Ø;
cmardind := 2Ø;

      (* type de fonte pour les elements de structure *)

cfonaute := italique;   cfondate := italique;   cfonenn1 := gras;
cfonenn2 := normal;     cfonenn3 := normal;     cfonenn4 := normal;
cfonlist := normal;     cfonnote := normal;     cfonfigu := italique;
cfonpara := normal;     cfonresu := normal;     cfontitr := gras;
                        cfontabl := normal;

    #       (* type d'implementation pour la liste *)

cimplist := tiret;

      (* type de police pour les elements de structure *)

cpolaut := 11;          cpoldat := 11;          cpolen1 := 14;
cpolen2 := 11;          cpolen3 := 11;          cpolen4 := 11;
cpollis := 11;          cpolnot := 1Ø;          cpolfig := 1Ø;
cpolpar := 11;          cpolres := 11;          cpoltit := 14;
                        cpoltab := 11;

    (* type de soulignement pour les elements de structure *)

csouaut := rien;        csoudat := rien;        csouen1 := continu;
csouen2 := continu;     csouen3 := continu;     csouen4 := continu;
csoulis := rien;        csounot := rien;        csoufig := rien;
csoupar := rien;        csoures := rien;        csoutit := rien;
                        csoutab := rien;

                    (* type de feuille *)

ctypefeuf := a4;

proportionblanc         := 1.5 ;
```

```
rewrite(res);
writeln('JE TRAVAILLE . . .');
writeln;
writeln('Si vous avez fait des erreurs dans le texte, en voici les numeros :');
D
writeln;
rewrite(resfor);rewrite(erreurs);
ARBRESTRUCTURE(adressepremierrecord);
 (* PARCOURSARBSTRUC(adressepremierrecord); *)
writeln('JE TRAVAILLE . . .');
ARBREFORMATAGE(adressepremiertypographique,adressepremierrecord);
 (* PARCOURSARBFORM(adressepremiertypographique); *)
writeln('JE TRAVAILLE . . .');
FDDTRAITEMENT(adresse,adressepremiertypographique);
writeln('JE TRAVAILLE . . .');
IMPRBIT(adresse);
test := false;
1 : if test then ERREUR(1);
writeln;
writeln('J''ai fini mon travail ');
end.
```

ANNEXE C

DESCRIPTION

DU

"FORMATTED DOCUMENT DESCRIPTION"

INTRODUCTION

Cette annexe contient la description du format "FDD" réalise a l'EPFL par ROGER HERSCH

FORMATTED DOCUMENT DESCRIPTION

R.D. Hersch

Laboratoire de Microinformatique
EPFL

CONTENTS

FORMATTED DOCUMENT DESCRIPTION (FDD)

1. Introduction

Todays high-resolution printers like VERSATECs electrostatic printers or laser printers are essentially graphic printers. Their technology makes it possible to print text of typographic quality, mixed with graphics and bitmap images. The aim of this paper is to define a standard representation of a formatted document containing text, graphics and bitmap images, which is easely understandable by a microprocessor driven printer controller. Such a standard will assure that several different formatters and editors can generate a document description which will be interpretated by control devices like a raster-scan display driver or by the final printer controller for printing..

Let us consider these two typical configurations:

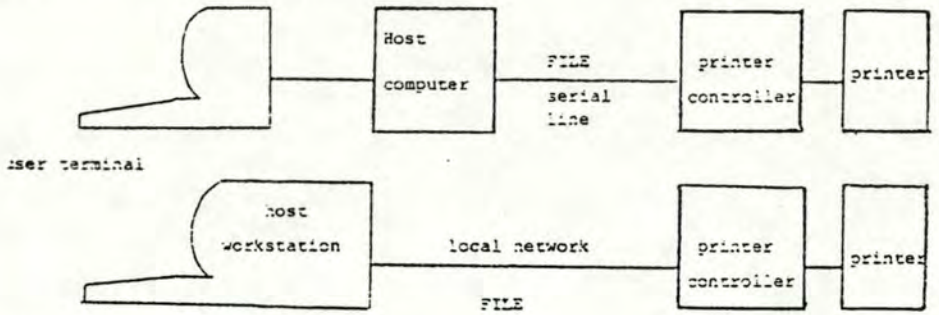


Fig. 1 Typical configurations

The division of work between host station or host computer and printer controller is important. Text, graphics and image processing and formatting programs [1] run on the host computer. They produce a formatted description of a document (FDD), alternatively called Raw Text File (RTF) or Formatted File. On the printable formatted file, each character, graphic primitive and bitmap image piece has its exact place, as well as it attributes (font, colours, mode, etc.) defined. The unit used for the coordinate system is one pixel, at the final printer's resolution.

2. Notations

In the Formatted Document Description (FDD), length of information and starting addresses are given in units of records (each 512 bytes) or units of bytes. Byte numbering and representation of numbers and characters in memory are Big-Endian.

Let us a look at a 32 bit integer in memory:

	For example	H' 0FF43CA0	and 'STRING'
increasing memory locations ↓			
byte address a	byte 0 byte 1	0 F F 4	S T
byte address a+2	byte 2 byte 3	3 C A 0	R I
			N G

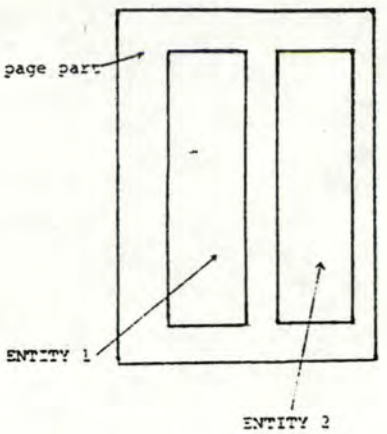
H' stands for hexadecimal number representation; D' stands for decimal representation

3. The Structure of the Formatted Document Description (FDD)

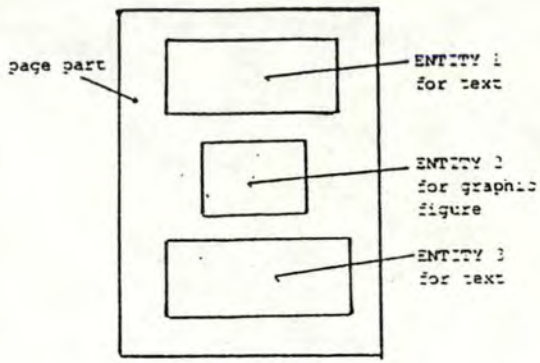
FDD is a formatted representation of a document to be printed. Such a document has the following structure : it is devided in pages (like in a real document). On each page there are several entities.

Examples:

a) page with text on 2 columns



b) page with text and graphics



At the start of the document, a PART DIRECTORY has an entry for each page part, defining the record location of the start of a page, as well as its length. Some other, special parts may be defined for external references, like the optional External File Directory Part. The whole document has a DOCUMENT HEADER containing general information about the document. In order to store the Formatted Document Description in a sequence of logical blocs on mass storage, it is devided in Records of fixed length (512 bytes). Each part as well as the Part Directory and Document Header has an integer number of records.
If the document has only one printed page, word 0 of the Document Header will be 0 and the description of the Single Formatted Page will follow immediately (byte address 129) after the Document Header

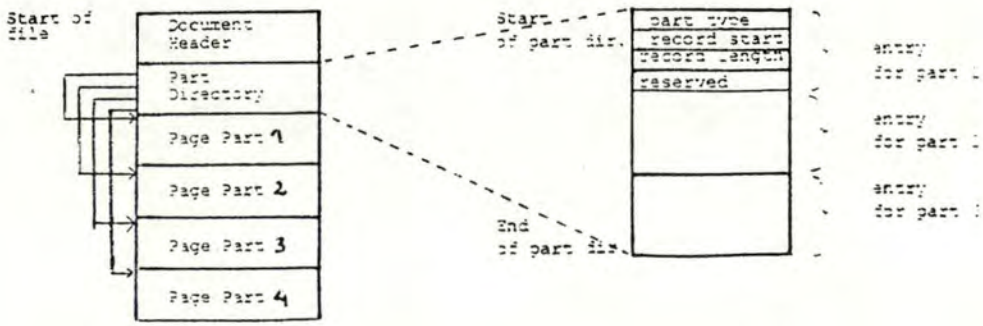


Fig. 2 Structure of a document in FDD format

4. DOCUMENT HEADER .

Document Header and Part Directory are within an integral number of records. The document Header is 128 bytes long and directly followed by the part directory.
For a single page formatted document description, the single page description follows immediately after the document header.

word 0	Must be (-1) for a Formatted Document Description Must be (0) for a Single Page Formatted Document Description
word 1	Total number of records in this file
word 2	Number of Parts (not including Part Directory)
word 3	Explicit record number where Part Directory begins, Implicitely: Part Directory is one record in front of Document Directory
word 4	Number of Records occupied by the Part Directory
word 5,6,7	Date given by 6 Ascii Characters (example: 170682)
word 8	First copy to print (else: -1)
word 9	Last copy to print (else: -1)
word 10	First page to print (else -1)
word 11	Last page to print (else -1)
word 12	Printing mode (else -1)
word 13	reserved
word 14	Final printer resolution in dots per inch
word 15	Page format: 3 for A4 0 for B4
word 14 to 28	26 bytes for string specifying FDD file name
word 29 to 41	25 bytes for string specifying the creator's name
word 42 to 63	reserved for comments

Word 12 allows the user to affect the appearence of the final document."R" code is given to print a mirror image of the defined document. "S" considers that all items are considered to be solid; each item overwrites the previous item. "T" indicates that all items, except <Show-dots-opaque> are to be considered as transparent.

5. PART DIRECTORY

The Part Directory together with the Document Header is an integer number of records. The Part Directory starts at byte 128 of FDD; its length is given in word 4 of the Document Header.

Each part is described by a four-word entry in the Part Directory:

word 0 Part type, interpreted as follows:

- 0: Part is a Printed Page
- 1: Part is the Optional Font Mapping Table Specifying Part, in which each entry contains a mapping table between font number and font described by family, style and size and rotation.
- 2: Part is the optional External File Directory Part. If FDD uses entities defined in other files, the names of the files requested must appear in the External Directory Part.
- 3: Part is the optional Graphic Pattern Directory Part; it contains an entry for each Graphic Pattern
- 4: Part is the Optional Special Character Sequence Definition Part
- <0: Part contains information used by higher level application programs. This information is not needed for output devices.

word 1 Record number, where part begins (First Record on FDD has number 0).

word 2 Length in records of the information for this part

word 3 reserved

6. The Optional Graphic Pattern Directory Part

Each entry defines a line or a surface pattern; they are numbered by their entry number in the Graphic Directory Part. Entry numbers start with entry number zero.

Format of each entry:

word 0: [entry length 2]

word 1: [code 2]

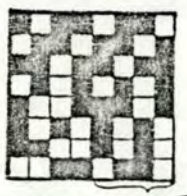
word 2: [s 1]

words following: [if line: dots-data s]
[if surface: dots-data s^2]

Specifies place taken by this entry in number of bytes
If code = 0 then a surface pattern is defined, else code indicates the line width of the correspondent line pattern.
The first byte of word 2 indicates how many bytes follow to describe the line pattern; if a surface pattern is described, s defines the side length of the quadratic pattern, in bytes.

From the second byte of word 2, the pattern is described, dotwise, a byte being a set of 8 dots.

example:



Hexadecimal representation

'01C''0000''04''B6''6DDB''559D''AACA''3FFF'

word 0 word 1 word 2 — hexadecimal pattern representation

Fig. 3. Description of a 8 x 8 bit surface pattern

7. The Optional Mapping Table Specifying Part

The default Mapping Table between font numbers and fonts defined by family, style, size and rotation (between 0 and 90 degrees) is known implicitely. If another Mapping Table is used, it has to be described in the Optional Mapping Table Directory Part.

Format of one entry in mapping table (5 bytes)

entry 0: [family-number 1] [style-number 1] [size in dots 2] [rotation in degrees 1]

This table has up to 4096 entries, ranging from entry 0 to entry 4095. The table is terminated by an entry consisting of 5 bytes with values (-1).

8. Optional External File Directory Part

This part contains the specifying name of all external files that are referred to in the document description (FDD). Each entry is 32 bytes long, byte zero specifies the number of valid characters in the string.

format:

ext-dir-entry 0: [external file name string of up to 31 characters)]
ext-dir-entry 1: [external file name string of up to 31 chararacters)]

9. The PRINTED PAGE PART

The printed page part contains all information for printing characters, graphics and bitmap image pieces. The printed page is divided into entities. Entities are defined by their bounding box. An entity describes a shunk of a page that belongs logically together : for example one paragraph is one entity, or a figure, or a table. Because all characters, graphics and dots are defined relatively to their entity bounding box, it is easy to move and place entities on the page. Entities defined in other documents may be referenced.

The entity enables the simultaneous display of typographic characters, graphics and bitmap image pieces. Therefore each entity is devided into 2 parts. The first part (ENTITY LIST) contains printing instructions for the printer subsystem. The second part (DATA LIST) contains the data to be printed.

Positionning as well as typographic commands for character printing are Entity List commands. Character strings to be printed are located in the Data List. For printing graphics and bitmap images, special Entity list commands like *<Interpreting Data List> <Graphics Mode> [n]* or *<Interpreting Data List> <Dot Image> [n]*, make the printing program interpreting the next n Data Bytes as Graphics printing commands or as Dots printing Commands. In future extensions, further arguments may be added, for example for interpreting Data List containing HP-Plotter commands.

The printed page part contains first the entity lists, then a 0 word and then the data lists. Because the Printed Page Part length is an integral number of records, the space from the end of the data list to the record end is padded with H'FF bytes.

Each entity list contains a byte pointer showing the beginning of the correspondant Data List, referenced from the start of the present page part.

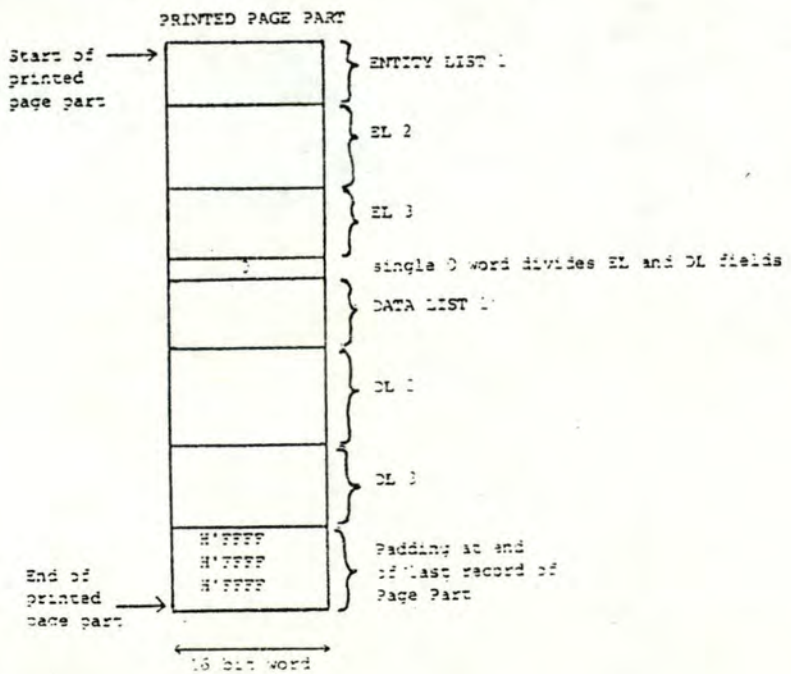


Fig. 4 Structure of a Printed Page Part

10. THE ENTITY LIST

The Entity List itself is divided in a HEADER part and in a COMMAND part. The Header contains all information relevant to the present Entity. The command part contains all Entity List commands.

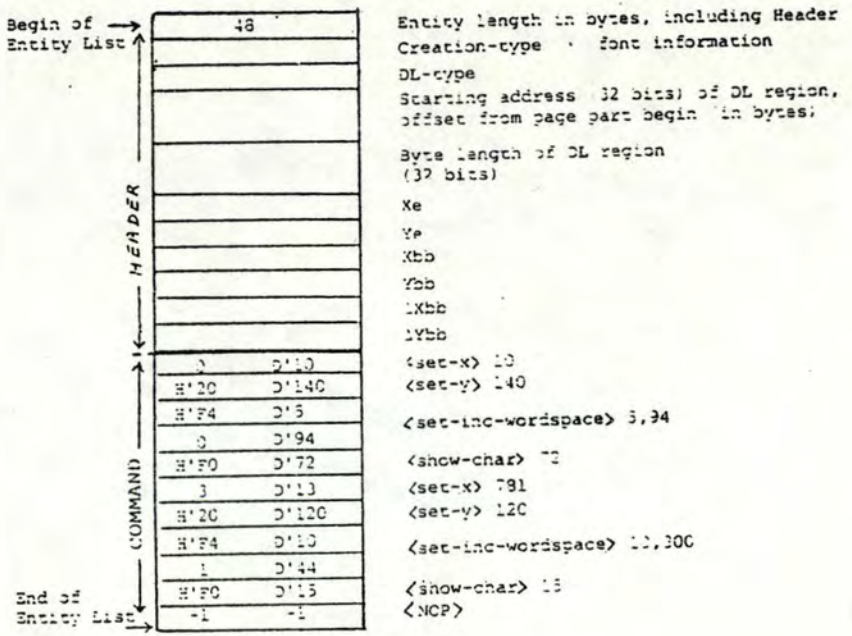
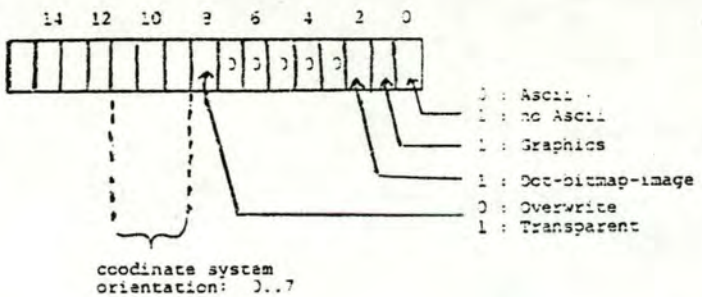


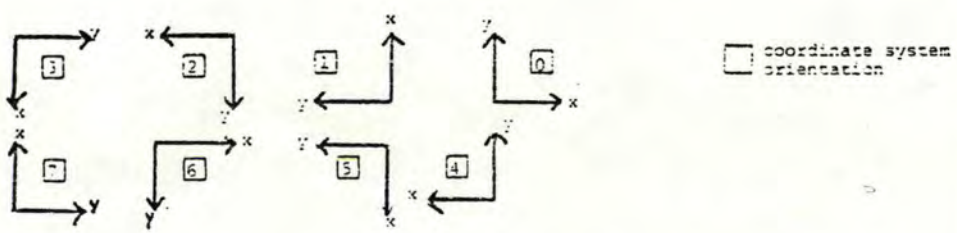
Fig. 5. A typical Entity List

10 a) FORMAT OF THE "ENTITY HEADER"

The associated number indicates the field length in bytes

[entity-length 2]	Entity length in bytes, including Entity Header
[creation-type 1]	indicates which program generated the entity
[font-set 1]	code specifying mapping table used for mapping font numbers in real character set defined by family, style and size
[DL-type 2]	Byte 0 indicates orientation of bounding box coordinates, as well as "overwrite" or "transparent" mode. Byte 1 indicates what kind of data is within correspondent DL; 0 : Data List contains ASCII characters only and Entity has a normally oriented coordinate system





[begin-byte 4]	Start of Data List (32 bits bytes address of starting position, relative to begin of present Page Part)
[byte-length 4]	Length of Data List (in bytes)
[Xe 2] [Ye 2]	Origin of coordinates system used for bounding box definition
[X_{bb} 2] [Y_{bb} 2] [ΔX_{bb} 2] [ΔY_{bb} 2]	These four parameters describe the bounding box of the entity relative to the origin of the coordinate system defined by (e,ye

10 b) Entity List Commands

Entity List commands indicate where to place and how to print characters, graphics and bitmap pieces. For text printing, the most important commands are positioning commands *<set-x>* and *<set y>*, a command defining the width of the "space" character, and orders to print a given number of characters, typically a line, in the previously defined mode.

When graphics is to be printed, the EL command *<Interpreting Data List> [Graphics Mode] [n]* specifies that the next n bytes in Data List are to be interpreted as graphic printing primitives.

For printing bitmap images, the EL command *<Interpreting Data List> [Dots Mode] [n]* specifies that the next n bytes in Data List are to be interpreted as bitmap dots image printing commands.

All coordinates are given in a coordinates system defined by the current bounding box of the entity.

<set-x b3:b0> +[x 1]	Sets current position; 0 <= x <= 4095
<set-y b3:b0> +[y 1]	Sets current position; 0 <= x <= 4095
<show-characters> [n 1]	The next n characters in DL are displayed at the current position (x,y). Current position is updated. For variable spaced text, space characters take predefined values given by <set-wordspace> command.
<show-EL-char> [char 1]	Prints character stored in argument EL byte
<show-sequence b2:b0> [n 1]	The n next characters form a special sequence to be interpreted and printed; the last 3 bits of the <show sequence> specify one of 8 possible interpretation tables. When all 3 bits are 1, the default interpretation is choosen; other interpretations used have to be specified in the Optional Special Character Sequence Definition Part (to be specified later).
<skip-characters> [n 1]	The n next bytes in DL are ignored; these bytes generally do not contain relevant information for the printer
<skip-bytes-in-DL> [type 1] [n 2]	Skips over control bytes in DL that may be needed for other purposes; type indicates what kind of program generated these control bytes
<skip-bytes-in-EL> [n 1]	Skips over the next n control bytes within EL
<fixed-spaced-char>	The following text has to be printed in fixed spaced character mode.
<variable-spaced-char>	The following text has to be printed in variable spaced character mode.
<Font> +[font-number 1]	This command changes the current font; it specifies another font (another style, size and family). That font is defined by its font number and by the table mapping font numbers to character sets defined by family, style and size and orientation.
<Set-inc-wordspace> [n 1] [s 2]	Defines width (s) of space character; on the first n spaces, the space used is the defined space incremented by one
<Set-wordspace b2:b0>+[x2]	<set wordspace>, 0 < n <2047, for all following space characters to be printed, the defined space is used without increments
<set-characterspace> [n 1]	Defines additional space between individual characters (default value: 0)

<fixed-space characters>

following text has to be printed in fixed spaced character mode

<variable-spaced character>

Following text has to be printed in variable spaced character mode, which is the default mode when nothing is specified

<highlighting> [parameter 1]

Specifies start or stop of highlighting action on characters (underline, overscore...).

<char-rotation> [rot 1]

Specifies 90, 180 or 270 degrees positive (counter-clockwise) rotation of the following characters. String direction is always given relative to the absolute character orientation. Character orientation is given by a character rotation value between 0 and 90 degrees defined within the character set and by the additional <char-rotation> parameter.
rot = 0 : No additional rotation of characters (default value)
rot = 1 : Rotation by 90 degrees, counter-clockwise
rot = 2 : Rotation by 180 degrees, counter-clockwise
rot = 3 : Rotation by 270 degrees, counter-clockwise

<string-direction> [dir 1]

Normally, (string-dir=0), a string is written orthogonally to the direction of its characters. If (string-dir=1), it will be generated, rotated by 90 degrees counterclockwise to its normal direction. Rotation is 180 degrees for (string-dir=2) and 270 degrees for (string-dir=3).

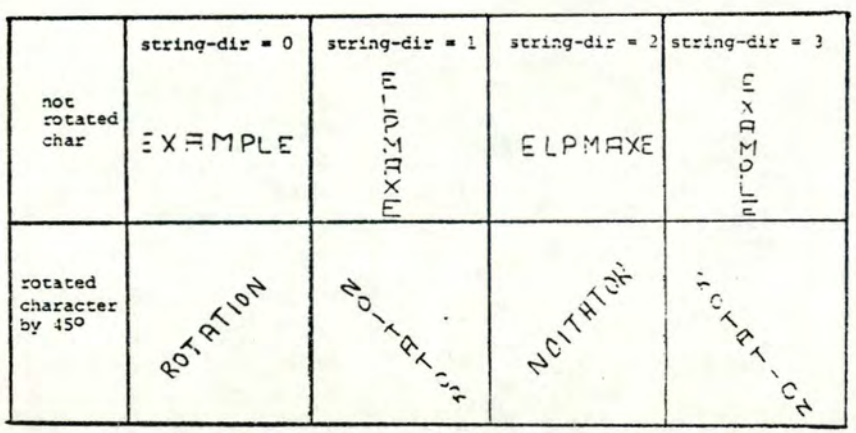


Fig. 6. Examples for illustration of <string direction>

<interpreting data list> [graphics mode 1] [n 2]

The next n bytes in DL are interpreted as a graphical object (see description of graphical commands in DL)

<interpreting-data-list> [transparent-dot-mode 1] [n 2]

The next n bytes in DL are interpreted as an image given by dots information. The image on the page is within a rectangle with its lower left corner at the current (x,y) position. Such a bitmap image is ORed with information generated previously. (See description of dots image commands in DL).

<interpreting-data-list> [opaque-dot-mode 1] [n 2]

Like <interpreting-data-list> [transparent-dot-mode], but the entire rectangular area specified is overwritten with the dots image pattern.

<only-on-copy>[n 1]

The following entity command is executed only on copy number n. It is skipped on other copy numbers. If n=0 the subsequent entity command is applied to all copies.

<get-entity-from-external-FDD-file> [entity number 1] [page-part 2] [ext-dir-entry 1] [nop 1]

specifies insertion of entity from external FDD file

11. DATA LIST

Each Data List belongs to an entity. In the Entity List Header, there is a pointer to the Data List. When representing text, the Data List contains a file of ASCII characters generally without formatting and structure indications. The Data List may optionally contain control information not destinated for printing purposes. When printing, these information is skipped by the printer subsystem, when interpreting the EL commands *<skip-bytes-in-DL>* or *<skip-characters>*.

The EL command *<interpreting-data-list> [graphics mode] [n]* means that the n next data bytes in DL are to be interpreted as graphic primitives printing commands. There are graphic primitive printing commands for line segments, circle segments, rectangles and polygons. Patterns for lines and surfaces may be user generated and stored in the Optional Graphic Pattern Directory Part.

The EL command *<interpreting-data-list> [dot-mode] [n]* specifies that the n next data bytes in DL are to be interpreted as encoded bitmap pieces (logos) to be printed dotwise.

11 a) Graphic commands in DL

Graphic commands do not move the current pointer within the entity. For each graphic command, a departure point (px,py) is given.

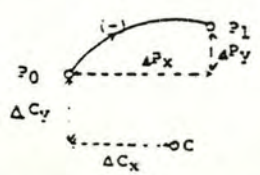
<<Draw-line>> [pattern-number 1] [px 2] [py 2] [Δx 2] [Δy 2]
Draws a straight line from point px,py to the next point given by Δx and Δy displacements (two's complement representation). The line is drawn with the pattern defined by its entry in the optional Graphic Pattern Directory Part; -1 specifies the default pattern.

<<Draw-rectangle>> [pattern-number 1] [px 2] [py 2] [Δx 2] [Δy 2]
Draws a rectangle from point px,py to the next x/y point given by Δx and Δy displacements. Lines are drawn with the pattern defined by its entry in the optional Graphic Pattern Directory Part; -1 specifies the default pattern

<<Fill-rectangle>> [pattern-number 1] [px 2] [py 2] [Δx 2] [Δy 2]
The area is filled with the pattern defined by its entry in the optional Graphic Pattern Directory Part; -1 specifies the default pattern.

<<Draw-circle-segment>> [orientation b7, pattern-number b6:b0 1] [p0x 2] [p0y 2] [Δcx 2] [Δcy 2] [Δpx 2] [Δpy 2]
Draws a circle segment given by the relative positions of center and destination from departure point (p0x,p0y). The line is drawn with the pattern defined by its entry in the optional Graphic Pattern Directory Part; -1 specifies the default pattern.

<<Pie>> [orientation b7, pattern-number b6:b0 1] [p0x 2] [p0y 2] [Δcx 2] [Δcy 2] [Δpx 2] [Δpy 2]
Fills a circle segment (pie) given by the relative positions of center and destination from departure point (p0x,p0y) with the predefined pattern. The area is filled with the pattern defined by its entry in the optional Graphic Pattern Directory Part; -1 specifies the default pattern.



Circle segment representation

<<Draw-polygon>> [nop 1] [pattern-number 1] [n 1] [px 2] [py 2] [Δx1 2] [Δy1 2]..[Δxn-1 2] [Δyn-1 2]
Draws Polygon with n corners given by departure point (px,py), and by the next points described by their relative displacements. The line is drawn with the pattern defined by its entry in the optional Graphic Pattern Directory Part; -1 specifies the default pattern.

<<Fill-polygon>> [nop 1] [pattern-number 1] [n 1] [px 2] [py 2] [Δx1 2] [Δy1 2]..[Δxn-1 2] [Δyn-1 2]
Like <<Draw-Polygon>>, but fills Polygon with previously defined surface pattern

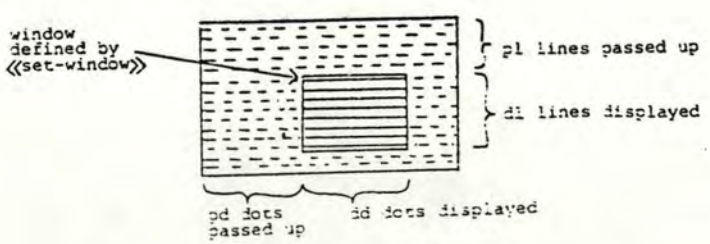
<<Get-graphics-from-external-file>> [ext-dir-entry 1]
Graphic commands (normally in DL) are stored on external file; pattern description is internal on FDD Graphic Pattern Directory Part

<<Get-graphics-from-external-FDD-file>> [ext-dir-entry 1] [page-part number 2] [entity number 1] [occurence number 1]
Graphic commands (normally in DL) are stored on external FDD file, at a given page part, entity and occurence number of graphic object description in DL; the correspondent pattern is in the external file, in its Graphic Pattern Directory Part

11b) Dots Image Objects:

<<set-coding>> [code 1] [columns 2] [lines 2]
This command specifies the window where dots-data will be displayed and specifies the encoding of these dots: "lines" gives the number of scanlines, "columns" the number of dots per scanline and "code" the encoding of the dots.
"Code" = 0
Bitmap Data: <Show-dots> generates transparent dots, when dot-data = 0; <Show-dots-opaque> generates non-intensified dots, when dot-data = 0 and both generate intensified dots when dot-data = 1
"Code" = 1
like "code"= 0 but inverted
"Code" = D'17
RUN-LENGTH encoded bitmap

<<Set-window> [nop 1] [pd 2] [dd 2] [pl 2] [dl 2]
gives window on which dots are displayed (they are clipped outside the window)



<<Set-mode>> [m 1]

Describes drawing direction of dots within lines and lines within rectangle.

default value: m = b b H'03
means dots from left to right
lines from top to bottom

Direction description code:

0 : from left to right --->
1 : from right to left <---
2 : from bottom to the top
3 : from top to the bottom

<<Set-relative-size>> [nop 1] [width 1] [height 1]
Default value: [width=1], [height=1]

Defines a relative factor in each direction between dots described on Raw Text File and dots of final printing device.

Example: [width =2, height=2] ==> One dot-data on RTF describes 4 dots on printer.

<<Dots-follow>>

The following bytes in DL contain the dots in the previously defined encoded form

<<Get-dots-from-file>> [ext-dir-entry 1]

The dots are stored in an external file. Name of the External File is stored in the optional External File Directory Part on the present FDD. Ext-dir-entry specifies the entry in which the name of the external file is listed. The content of this external file is treated as if it had been referenced in DL using <<Dots-follow>>

<<Get-dots-from-external-FDD>> [ext-dir-entry 1] [page-part number 2] [entity number 1] [occurence number 1]

Dots are assumed to be in DL of specified page of specified external FDD. The nth occurence of <interpreting DL> [dots-mode] is looked for and the correspondent DL bytes are interpreted.

12. COMMAND DEFINITIONS

All command opcodes are 1 byte wide

12 a) Entity list commands:

<set-x b4:b0> +[n 1]	H'00 to H'1F
<set-y b4:b0> +[n 1]	H'20 to H'3F
<set-wordspace b2:b0> + [n 1]	H'60 to H'67
<show-sequence b2:b0> [n]	H'68 to H'6F
<Font b3:b0> +[n]	H'70 to H'7F
<reserved>	H'80 to H'9F
<interpreting-data-list> [graphics-mode 1] [n 2]	<H'A0> [H'00]
<interpreting-data-list> [transparent-dot-mode 1] [n 2]	<H'A0> [H'01]
<interpreting-data-list> [opaque-dot-mode 1] [n 2]	<H'A0> [H'02]
<..spare..>	H'A1 to H'E3
<get-entity-from-external-FDD> [entity 1] [page part 2] [ext-dir-entry 1] [nop 1]	H'E4
<highlighting> [n 1]	H'E5
<fixed-spaced char>	H'E6
<variable-spaced char>	H'E7
<set-characterspace> [n 1]	H'E8
<char-rot> [n 1]	H'E9
<string-direction> [dir 1]	H'EA
<Skip-bytes-in-EL> [n 1]	H'EB
<Only-on-copy> [n 1]	H'ED
<Show-characters> [n 1]	H'F0
<Skip-characters> [n 1	H'F1
<Skip-bytes-in-DL> [type 1] [n 2]	H'F2
<Show-EL-character> [char 1]	H'F3
<Set-inc-wordspace> [n 1] [s 2]	H'F4
<Nop>	H'FF
reserved for <Alternative>	H'EC
reserved for <Set-x-long> [nop 1] [n 2]	H'EE
reserved for <Set-y-long> [nop 1] [n 2]	H'EF
reserved for <Set-brightness>	H'F8
reserved for <Set-hue>	H'F9
reserved for <Set-saturation>	H'FA

12 b) Data List commands:

a. for Graphics

<<Draw-line>> [pattern-number 1] [px 2] [py 2] [Δ x 2] [Δy 2] H'02

<<Draw-rectangle>> [pattern-number 1] [px 2] [py 2] [Δx 2] [Δy 2] H'03

<<Fill-rectangle>> [pattern-number 1] [px 2] [py 2] [Δx 2] [Δy 2] H'04

<<Draw-circle-segment>> [orientation-bit-b7, pattern-number b6:b0 1] [p0x 2] [p0y 2] [Δcx 2] [Δcy 2] [Δpx 2] [Δpy 2] H'05

<<pie>> [orientation b7, pattern-number b6:b0 1] [p0x 2] [p0y 2] [Δcx 2] [Δcy 2] [Δpx 2] [Δpy 2] H'06

<<Draw-polygon>> [nop 1] [pattern-number 1] [n 1] [px 2] [py 2] [Δx1 2] [Δy1 2]..[Δxn-1 2] [Δyn-1 2] H'07

<<Fill-polygon>> [nop 1] [pattern-number 1] [n 1] [px 2] [py 2] [Δx1 2] [Δy1 2]..[Δxn-1 2] [Δyn-1 2] H'08

<<Get-graphics-from-external-file>>[ext-dir-entry 1] H'08

<<Get-graphics-from-external-FDD-file>>[ext-dir-entry 1] [page-part number 2] [entity number 1] [occurence number 1] H'09

b. For Dot-patterns

<<set-coding>>[code 1] [dots 2] [lines 2] H'01

<<Set-window> [nop 1] [pd 2] [dd 2] [pl 2] [dl 2] H'02

<<Set-mode>>[m 1] H'03

<<Set-relative-size>> [nop 1] [width 1] [height 1] H'04

<<Dots-follow>> H'05

<<Get-dots-from-file>>[ext-dir-entry 1] H'06

<<Get-dots-from-file>>[ext-dir-entry 1] H'07

<<Get-dots-from-external-FDD>>[ext-dir-entry 1] [page-part number 2] [entity number 1] [occurence number 1] H'08

ANNEXE D

PROGRAMMES

DE

DECODAGE/ENCODAGE

INTRODUCTION

Cette annexe se compose de :

- les spécifications et le texte PASCAL d'un programme de décodage de format "BINAIRE" en format "S"
- les spécifications et le texte PASCAL d'un programme d'encodage de format "S" en format "BINAIRE"
- la description du format "S"
- un exemple de fichier source suivi de sa codification en format "S" et en format "SSM" (le format "SSM" n'est pas décrit, il est seulement donné à titre d'exemple)
- les commandes d'utilisation des programmes d'encodage et de décodage

Les spécification des programmes de décodage-encodage ne reprennent que l'algorithme logique ; pour plus de détails, se référer aux commentaires insérés dans le texte PASCAL de chaque programme.

1. SPECIFICATIONS DU PROGRAMME DE DECODAGE "BINAIRES"

Algorithme logique :

```
begin
  ouverture des fichier en input/output;
  initialisation des variables ;
  tant que pas fin de fichier input
    begin
      lecture de 8 bits du fichier source ;
      appel à la procédure BINHEXA2CAR qui recevant un byte, rend
      sa valeur hexadécimale en deux caractères ;
      appel à la procédure HEXADECI2CAR qui recevant deux caractères,
      rend leur valeur décimale ;
      calcul du BYTE COUNT ;
      mise à jour DU CHECKSUM ;
      gestion des 4 bits d'adressage ;
      gestion des types (S0,S1,S2,S9) ;
      création du CHECKSUN ;
      réinitialisation des variables ;
    end
end
```

2. TEXTE PASCAL DU PROGRAMME

```
(*****************************************************************************)

program SBINAIRE(r1Øhwh,r1Ø);

  (* programme ayant comme but de creer un fichier en r1Ø binaire de nom:r1Ø
     en partant d'un fichier en r1Ø s de nom :r1Øhwh *)

label 1;
type zeroun = Ø..1;
     byte = packed array[Ø..7] of zeroun;
var bete : byte;
    compteur,var1,var2,j,k,resul : integer;
    intermedi : zeroun;
    car,cara1,cara2 : char;
    r1Ø,r1Øhwh : text;

(*****************************************************************************)

procedure decode(var baite:byte;chiffre:integer);

  (* procedure ayant comme but de decoder un entier CHIFFRE, et de lui donner
  une valeur binaire (binaire a huit positions correspondant a un byte), le
  resultat se trouvant dans BAITE *)

var i : integer;

begin
  i := Ø;
  while i <= 7 do
    begin
      baite[i] := Ø;
      i := i + 1
    end;
  if chiffre - 128 >= Ø then
    begin
      chiffre := chiffre - 128;
      baite[Ø] := 1
    end;
  if chiffre - 64 >=Ø then
    begin
      chiffre := chiffre - 64;
      baite[1] := 1
    end;
  if chiffre - 32 >= Ø then
    begin
      chiffre := chiffre - 32;
      baite[2] := 1
    end;
  if chiffre - 16 >= Ø then
    begin
      chiffre := chiffre - 16;
      baite[3] := 1
    end;
  if chiffre - 8 >= Ø then
    begin
      chiffre := chiffre - 8;
      baite[4] := 1
    end;
  if chiffre - 4 >= Ø then
    begin
      chiffre := chiffre - 4;
      baite[5] := 1
    end;
  if chiffre - 2 >= Ø then
    begin
      chiffre := chiffre - 2;
      baite[6] := 1
    end;
  if chiffre - 1 >= Ø then
    begin
      chiffre := chiffre - 1;
      baite[7] := 1
    end
end;

(*****************************************************************************)

procedure INTGCAR(var intgcarcara : char;intgcarentier : zeroun);

  (* procedure ayant comme but de transformer un entier de type Ø..1
     INTGCARENTIER , en un caractere de meme valeur dans INTGCARCARA *)

begin
  if intgcarentier = Ø then intgcarcara := 'Ø';
  if intgcarentier = 1 then intgcarcara := '1'
end;

(*****************************************************************************)
```

```
procedure HEXADEC(var resultat:integer;var1,var2:integer);

  (* procedure ayant comme but de donner la valeur en decimal d'un nombre
     hexadecimal , on recoit deux entiers VAR1,VAR2 ou var1 est de poids le
     plus fort et on rend un entier RESULTAT *)


begin
  resultat := var1 * 16;
  resultat := resultat + var2
end;
{*************************************************************************}

procedure VALEUR(var numero:integer;c:char);

  (* procedure permettant de faire correspondre un element de type
     caractere C ,representation d'un nombre hexadecimal en un element de
     type entier NUMERO *)
begin
   case c of
          'Ø' : numero := Ø;
          '1' : numero := 1;
          '2' : numero := 2;
          '3' : numero := 3;
          '4' : numero := 4;
          '5' : numero := 5;
          '6' : numero := 6;
          '7' : numero := 7;
          '8' : numero := 8;
          '9' : numero := 9;
          'A' : numero := 1Ø;
          'B' : numero := 11;
          'C' : numero := 12;
          'D' : numero := 13;
          'E' : numero := 14;
          'F' : numero := 15
       end;
 end;
{*************************************************************************}


begin (* PROGRAMME PRINCIPAL*)

  (* programme de decodage,
  ce programme ouvre un fichier en input : fichier en r1Ø s
                                  en output : fichier en r1Ø binaire
  le but est de lire tous les caracteres du fichiers S et de donner
  un fichier de donnee en binaire
  NB . ce programme ne transforme que les donnees ou l'element d'identification

  est S1 (24 elements  de donnees par ligne);
  on pourrait faire un test initial pour chaque ligne afin de pouvoir considerer


  les autres possibilites et d'initialiser une variable afin de donner la valeur


  du nombre d'elements a prendre en consideration
  NB2. Cette procedure ne teste pas le cheksum *)

  compteur := 1;
  reset(r1Øhwh);
  rewrite(r1Ø);
  1 : read(r1Øhwh,cara1);
  read(r1Øhwh,cara2);
  while (not EOF(r1Øhwh)) do
  begin
    if ((cara1 = 'S') and (cara2 = '1')) then
      begin
        read(r1Øhwh,cara1);
        read(r1Øhwh,cara2);
        valeur(var1,cara1);
        valeur(var2,cara2);
        hexadec(resul,var1,var2);
        resul := resul - 1;
        j  := 1;
        while j <= 2 do
           begin
             read(r1Øhwh,CARA1);
             read(r1Øhwh,CARA2);
             resul := resul - 1;
             j := j + 1
           end;
        k := resul ;
        while k  > Ø do
           begin
             read(r1Øhwh,cara1);
             read(r1Øhwh,cara2);
             valeur(var1,cara1);
             valeur(var2,cara2);
             hexadec(resul,var1,var2);
             decode(bete,resul);
             compteur := compteur + 1;
             j := Ø;
```

```
            while j <= 7 do
              begin
                intermedi := bete[j];
                intgcar(car,intermedi);
                write(r10,car);
                j := j + 1
              end;
            writeln(r10);
            k := k - 1
          end
      end
    else
    begin
      goto 1
    end
  end
end.
```

3. SPECIFICATION DU PROGRAMME "SBINAIRE"

Algorithme logique :

```
begin
  ouverture des fichiers en input/output ;
  tant que pas fin de fichier input
    begin
      lecture de l'élément d'identification (S0,S1,S2,S9) ;
      lecture de deux caractères hexadécimaux ;
      appel à la procédure VALEUR qui recevant la valeur entière
      de deux caractères lus, la transforme en une valeur décimale ;
      appel à la procédure ENCODE qui recevant une valeur décimale
      l'encode sous forme de byte ;
      impression du byte ;
    end
end
```

4. TEXTE PASCAL DU PROGRAMME

```
(*************************************************************************)

programbinaires(defini,output,suivant);

  (* le but de ce programme est de transformer un fichier binaire,
     de nom defini.dat  en un fichier de for mats, de nom suivant.dat *)

label 1;
type zeroun = Ø..1;
     byte = packed array[Ø..7] of zeroun;
     typetampon = array[Ø..8Ø] of char;
var intermedi,by : byte;
    defini,suivant : text;
    reste,test : boolean;
    cheksum,dec,compteurdefini,i,valeurentiere,adressecourante,
    position,compteurbyte,indice : integer;
    resultat1,resultat2,ad1,ad2,ad3,ad4,hex1,hex2 : char;
    tampon : typetampon;
    caractere : char;

(*************************************************************************)

function DECIHEXA1CAR(entier :integer):char;

  (* fonction qui recevant un integer :enter, en rend sa valeur hexadecimal
     sous forme de caractere :DECIHEXA1CAR *)

begin
   case  entier of
         Ø :DECIHEXA1CAR := 'Ø';
         1 :DECIHEXA1CAR := '1';
         2  :DECIHEXA1CAR:= '2';
         3 :DECIHEXA1CAR := '3';
         4 :DECIHEXA1CAR := '4';
         5 :DECIHEXA1CAR := '5';
         6 :DECIHEXA1CAR := '6';
         7 :DECIHEXA1CAR := '7';
         8 :DECIHEXA1CAR := '8';
         9 :DECIHEXA1CAR := '9';
         1Ø :DECIHEXA1CAR := 'A';
         11 :DECIHEXA1CAR := 'B';
         12 :DECIHEXA1CAR := 'C';
         13 :DECIHEXA1CAR := 'D';
         14 :DECIHEXA1CAR := 'E';
         15 :DECIHEXA1CAR := 'F'
       end;
end;

(*************************************************************************)

procedure BINHEXA2CAR(var hex1,hex2 : char; byy : byte);

  (* procedure qui recevant un byte- tableau de Ø..7 of Ø..1-,byy, rend sa
     valeur hexadecimal sous la forme de 2 caracteres: hex1,hex2 ou hex1 est
     l'element  de poids le plus fort.
     exemple : ØØØØØ11Ø = 4 en binaire = 4 en hexadecimal : valeur Ø4 *)

var  i,entier : integer;

procedure BINAIREDECIMAL(var ent : integer;bi : byte);extern;

procedure DECIHEXA2CAR(var h1,h2 : char; ent : integer);extern;

begin
  binairedecimal(entier,byy);
  decihexa2car(hex1,hex2,entier);
end;

(*************************************************************************)

procedure BINAIREDECIMAL(var entier :integer;byy : byte);

  (* procedure qui recevant un byte de huit bits ,byy, renvoit sa valeur entiere

     decimal dans entier*)

var somme: integer;

begin
  somme := Ø;
  if byy[Ø] = 1 then somme := somme + 128;
  if byy[1] = 1 then somme := somme + 64;
  if byy[2] = 1 then somme := somme + 32;
  if byy[3] = 1 then somme := somme + 16;
  if byy[4] = 1 then somme := somme + 8;
  if byy[5] = 1 then somme := somme + 4;
  if byy[6] = 1 then somme := somme + 2;
  if byy[7] = 1 then somme := somme + 1;
  (* dans la variable somme, on  a la valeur en binaire du byte *)
  entier := somme;
end;

(*************************************************************************)
```

```
procedure DECIHEXA2CAR (var hex1,hex2 : char; somme : integer);

  (* procedure recevant un entier decimal,somme, le transforme en deux caractere
hexadecimaux hex1 et hex2 ou hex1 est de poids le plus fort
au cas ou la valeur passee est 255, la procedure rendra la valeur FF *)

var redi,ef,zero,reste : integer;
function DECIHEXA1CAR(entier :integer):char;extern;

begin
  zero := 0;
  ef := 15;
  if somme = 255 then
    begin
      hex1 := DECIHEXA1CAR(ef);
      hex2 := DECIHEXA1CAR(ef);
    end
    else
    begin
      if ((somme / 16) >= 1) then
        begin
          redi := (somme div 16);
          hex1 := DECIHEXA1CAR(redi);

         (* calcul du reste, on a la valeur entiere dans redi *)

          reste := somme - (redi * 16);
          hex2 := DECIHEXA1CAR(reste);
        end
        else
        begin
          hex1 := DECIHEXA1CAR(zero);
          hex2 := DECIHEXA1CAR(somme);
        end
    end
end;

(*****************************************************************************)

procedure HEXADECI1CAR(var decimal: integer;hexadecimal : char);

  (* procedure recevant un caractere hexadecimal, HEXADECIMAL rend sa valeur
     decimal DECIMAL *)
begin
  case hexadecimal of
                  '0' :decimal := 0;
                  '1' :decimal := 1;
                  '2' :decimal := 2;
                  '3' :decimal := 3;
                  '4' :decimal := 4;
                  '5' :decimal := 5;
                  '6' :decimal := 6;
                  '7' :decimal := 7;
                  '8' :decimal := 8;
                  '9' :decimal := 9;
                  'A' :decimal := 10;
                  'B' :decimal := 11;
                  'C' :decimal := 12;
                  'D' :decimal := 13;
                  'E' :decimal := 14;
                  'F' :decimal := 15;
                end;
end;

(*****************************************************************************)

procedure HEXADECI2CAR(var decrchek : integer; decrchek1,decrchek2 : char);

  (* procedure recevant deux caracteres hexadecimaux, DECRCHEK1,DECRCHEK2 ou
     DECRHEK est de poids le plus fort rend leur valeur decimal,DECRCHEK *)


var decrvar1,decrvar2 :integer;

procedure HEXADECI1CAR(var en : integer; caractere : char);extern;

begin
  HEXADECI1CAR(decrvar1,decrchek1);
  HEXADECI1CAR(decrvar2,decrchek2);
  decrchek := ((decrvar1 * 16) + decrvar2);
end;

(*****************************************************************************)
```

```
procedure GESTIONADRESSE(var gd1,gd2,gd3,gd4 : char; gdcouradr : integer);

  (* recevant un entier : GDCOURADR, on le transforme en hexadecimal,et on
   met dans GD1,GD2,GD3,GD4 ou gd1 est l'element de poids le plus fort *)

var gdentier : integer;

begin
  gd1 := 'Ø';
  gd2 := 'Ø';
  gd3 := 'Ø';
  gd4 := 'Ø';
  if ((gdcouradr / 4Ø96) >= 1 ) then
    begin
      (* cas ou la valeur de gdcouradr vaut 16 * 16 * 16  au moins *)

      gdentier := (((gdcouradr div 16)div 16) div 16);
      gd1 := DECIHEXA1CAR(gdentier);
      gdcouradr := gdcouradr - (((gdentier * 16) *16 ) * 16);
    end;
  if ((gdcouradr / 256 ) >= 1) then
    begin
      gdentier := ((gdcouradr div 16)DIV 16);
      gd2 := DECIHEXA1CAR(gdentier);
      gdcouradr := gdcouradr - ((gdentier * 16) * 16 );
    end;
  if ((gdcouradr / 16) >= 1) then
    begin
      gdentier := (gdcouradr div 16);
      gd3 := DECIHEXA1CAR(gdentier);
      gdcouradr := gdcouradr - (gdentier * 16);
    end;
  gd4 := DECIHEXA1CAR(gdcouradr)
end;
(*****************************************************************************)

  (* procedure de decodage binaire - format s
     Cette procedure ouvre un fichier en input dans lequel se trouve le format

     binaire et ouvre un deuxieme fichier en output pour y mettre le resultat
       du decodage.
     La premiere ligne du fichier consiste en une ligne d'entete (etiquette
     SØ) les donnees suivent (24 donnees par ligne) et le fichier se termine
     par un (S9) *)

begin
  reset(defini);
  rewrite(suivant);

  (* impression du header du fichier *)

  writeln(suivant,'SØØØØØØØ738C');

  (* impression des datas *)

  reste := true;
  cheksum := Ø;
  adressecourante := Ø;
  compteurdefini := Ø;
  compteurbyte := 1;
  position := 9;
  while not eof(defini) do
    begin
      while not eoln(defini) do
        begin
          read(defini,caractere);
          if compteurdefini < 8 then
            begin

              (* memorisation du bit dans la position courante de by *)

              if caractere = 'Ø' then  by[compteurdefini] := Ø;
              if caractere = '1' then  by[compteurdefini] := 1;
              compteurdefini := compteurdefini + 1;
            end
            else
        1: begin
             if compteurbyte <= 24 then
               begin
                 binhexa2car(hex1,hex2,by);
                 tampon[position] := hex1;
                 position := position + 1;
                 tampon[position] := hex2;
                 position := position + 1;
                 HEXADECI2CAR(dec,hex1,hex2);
                 cheksum :=cheksum + dec;
                 compteurbyte := compteurbyte + 1;

                 (* au cas ou 1' on ne passe pas dans la boucle suivante
                    on memorise le caractere lu dans la premiere position
                    du byte *)

                 if compteurbyte <= 24 then
                   begin
                     if caractere = 'Ø' then by[Ø] := Ø;
                     if caractere = '1' then by[Ø] := 1;
                     compteurdefini := 1;
                   end;
               end;
             if ( (compteurbyte > 24) OR not reste) then
               begin
```

```
                    (* imprimer le byte count *)

                    valeurentiere := Ø;
                    hex1 := DECIHEXA1CAR(Ø);
                    compteurbyte := compteurbyte + 2 ;
                    if ((compteurbyte / 16 ) >= 1) then
                      begin
                        valeurentiere := (compteurbyte div 16);
                        hex1 := DECIHEXA1CAR(valeurentiere);
                        compteurbyte := compteurbyte - (valeurentiere* 16);
                      end;
                    hex2 := DECIHEXA1CAR(compteurbyte);
                    tampon[3] := hex1;
                    tampon[4] := hex2;
                    cheksum := cheksum + compteurbyte +(valeurentiere * 16);

                    (* gestion des adresses *)

                    GESTIONADRESSE(ad1,ad2,ad3,ad4,adressecourante);
                    tampon[5] := ad1;
                    tampon[6] := ad2;
                    tampon[7] := ad3;
                    tampon[8] := ad4;
                    HEXADECI2CAR(dec,ad1,ad2);
                    cheksum := cheksum + dec;

                    HEXADECI2CAR(dec,ad3,ad4);
                    cheksum := cheksum + dec;

                   (* gestion du type *)

                    tampon[1] := 'S';
                    tampon[2] := '1';
                    adressecourante := adressecourante + 24;

                    (* creation du chek sum *)

                    cheksum := (255 - (cheksum mod 256));
                    decihexa2car(hex1,hex2,cheksum);
                    tampon[position] := hex1;
                    position := position + 1;
                    tampon[position] := hex2;

                    (* on incremente l'adresse du nombre d'element *)

                    indice := 1;
                    while indice <= position  do
                      begin
                        write(suivant,tampon[indice]);
                        indice := indice + 1;
                      end;
                    writeln(suivant);

                    (* reinitialisation *)

                    compteurbyte := 1;
                    position := 9;
                    cheksum := Ø;

                    (* cas ou on a lu un caractere et qu'on a du traiter
                       les huit caracteres precedents *)

                    if caractere = 'Ø' then by[Ø] := Ø;
                    if caractere = '1' then by[Ø] := 1;
                    compteurdefini := 1;
                  end;
              end;
          end;
        if reste then readln(defini);
      end;

    (* impression de la fin du fichier *)

    if ((position > 9) and reste ) then
      begin
        reste := false;
        goto 1
      end;

      (* Etant donne que la derniere ligne du fichier n'est peut-etre
         pas traitee dans le cas ou position est superieure a 9, on force
         le traitement de cette ligne, de plus si la derniere ligne est
         complete on n'a pas encore traite le dernier caractere, donc on
         force egalement se traitement  *)

  write(suivant,'S9Ø3ØØØØFC');
end.
```

5. DESCRIPTION DU FORMAT "S"

S RECORDS

An S record is a standard format used in transmitting and receiving programs and data.

There are ten possible standard S record types, six of which are in active use, two are defined but are not in active use, and two are reserved. They are as follows:

S0	Header record	active
S1	16 bit address Data record	active
S2	24 bit address Data record	active
S3	32 bit address Data record	
S4	reserved	
S5	Transmitted Data Record Count record	active
	reserved	
S7	32 bit address End of File/Execution Address record	
S8	24 bit address End of File/Execution Address record	active
S9	16 bit address End of File/Execution Address record	active

The standard S record is defined as follows:

Frame	Value	Description	Byte Counted	Check Summed
	$0D	Carriage Return		
	$0A	Line Feed		
	$00	Null		
1	$53 (S)	Start of Record		
2	$30-$39 (0-9)	Record Type		*
3,4		Byte Count	*	*
5-8		Address (for 16 bit)	*	*]
[5-10		Address (for 24 bit)	*	*]
[5-12		Address (for 32 bit)	*	*
:			*	*
:		Data	*	*
:			*	
N-1,N		Checksum		

"S" and the record type are represented directly in ASCII.

The byte count, address, data, and checksum are represented in ASCII encoded hexadecimal; i.e., two frames per data byte, with the most significant digit in the leading frame.

The checksum is the 1's complement of the sum of all 8-bit data/address bytes (not frames) from byte count to last data byte, inclusive.

The number of data bytes typically transmitted is either 24 or 16; at two frames per data byte, the S record length for 16 bits is N = 58 or 42 frames, respectively. This includes start of record (1), record type (1), byte count (2), checksum (2), address (4 for 16 bit). The maximum S record length N=70 frames.

For S0: Address = 0000
Data = header of file or message

S1,S2,S3: Address = location for first byte of record
Data = data to be stored

S5: Address = Transmitted Data Record Count (16 bits)
Data = none

S7,S8,S9: Address = Program Execution Entry Address
(=0000 if no Entry Address)
Data = none

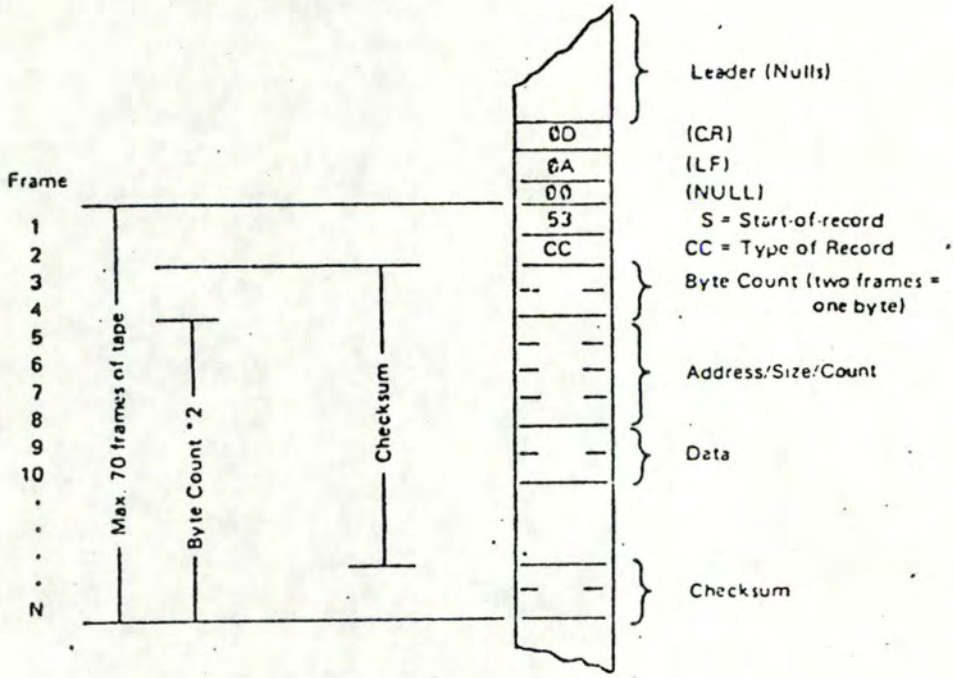
THE Transmitted Data Record Count of S5 is the count of all S1, S2, and S3 records transmitted. It is used to measure the quality or completeness of transmission by comparing with the Count of Records actually received.

A file transmission is typically made up of the following sequence of records/components:

1. Leader
2. S0 — Header record
3. S1, S2 or S3 — Data records
4. S5 — Transmitted Record Count record
5. S9 — End of File/Execution Entry Address record
6. Trailer

The leader is a string of a minimum of 32 nulls followed by a carriage return, line feed and null sequence.

The trailer is a string of a minimum of 32 nulls.



Frames 3 through N are hexadecimal digits represented by a 7-bit ASCII character. Two hexadecimal digits are combined to make one 8-bit byte.

The checksum is the one's complement of the summation of 8-bit bytes.

Frame	CC = 30 Header Record		CC = 31 Data Record		CC = 35 Record Count Record		CC = 39 End-of-File Record	
1. Start-of-Record	53	S	53	S	53	S	53	S
2. Type of Record	30	0	31	1	35	5	39	9
3. Byte Count	31	12	31	16	30	03	30	03
4. Byte Count	32		36		33		33	
5	30		31		30		30	
6 Address/Size/Count	30	0000	31	1100	31	0123	30	0000
7. Address/Size/Count	30		30		32		30	
8.	30		30		33		30	
9. Data	34	48 H	39	98	46	FA	46	FC
10. Data	38		38		41		43	
	34	44-D	30	02	(Checksum)		(Checksum)	
	34		32					
	35	52-R						
	32		38	8A (Checksum)				
			41					
	39	9E						
N. Checksum	45							

Tape Format for S Records

TYPICAL OBJECT OUTPUT S-RECORD FORMAT

```
S00600004844521B
S1131D0C3D7C10D0327C1FFE23C0D3042BC478D00
S1131010383C09964A016A000012A18B0C96600E1
S1131020000AD2FC00026000002EE3113400E352F7
S11310300242000BE30D0504660000006E25860D48A
S1131040E2580840000F60CC4A016A00000A1A18EE
S1131050B0C96700002AE3113400E3520242000BD6
S11310600504660000000E35B08C300006000000390
S1131070E35B0883000E25808C0000F60CA31C374
S107108001FFE1E728B
S00600004844521B
S20A010000323C00035641ED
S9030000FC
READY
```

First two characters
- S0 Starts output of the first record.
- S1 Indicates that the object data that follows will be at a two-byte memory address.
- S2 Same as S1, but indicates a three-byte memory address.
- S9 Last record.

Third and fourth characters - Hexadecimal byte count of the remaining characters in the record.

Fifth through eighth characters - Hexadecimal memory address where the data that follows is to be loaded. If the record is "S2" type, the fifth through tenth characters contain the memory address.

Last two characters - Checksum of all characters from byte count to the end of data.

6. EXEMPLE GLOBAL

6.1 Fichier source du texte au kilomètre

```
\\DE:P:10
\\INT:RE: 2.
\\FO:RE:ITA.
\\PO:RE:11.
\\PO:NO:14.
\\SO:EN:N:3: C.
\\PA:S.
\\TA:3.
\\UN.
\DO.
\TI.8.3.2. "MAUVAIS" EXEMPLE
\AU.Mathieu - Schmitz & David Levy
\DA.Lausanne, 12 - avril - 1983
\RE.Le texte source se compose de caract\'eres constituant les commandes
  du niveau global, les commandes du niveau local et le texte.
\\CR.Le contenu du texte n'a aucune
  importance. Les commandes du niveau global sont les
  commandes situ\'ees avant la s\'equence\\SOU : C. <\DOCU.>.\\FIN:S. Elles
  ont \'et\'e choisies afin de pr\'esenter les diff\'erentes commandes
  de formatage ; elles n'ont pas \'et\'e choisies pour obtenir une
  pr\'esentation convenable du texte .
\\CR.La liste des commandes n'est pas exhaustive ; nous trouverons des
  commandes de niveau globale et des commandes de niveau local.
\\CR.Parmi les
\\SOU : DI.commandes du niveau global\\FIN : SO., nous trouverons,
par exemple, un num\'erotatage de pages du document
  - num\'erotage en haut de page, une table des mati\'eres \'a
  trois niveaux,
  , une police num\'ero 14 pour les notes bas de page,
  un interligne de 2 millim\'etres pour l' \'el\'ement de type
  r\'esum\'e, les \'el\'ements d'identifications du document sur
  une premi\'ere page, centr\'es et non num\'erote\'es.
\\CR.Parmi les \\SOU : DI.commandes de niveau local\\FIN : SO. -
modifiant le formatage de l'\'el\'ement logique courant,
  nous trouverons, par exemple, la sp\'ecification d'une fonte, d'une police,
  et d'un soulignement diff\'erent de celui d\'efini soit au niveau global, soit
  pr\'ede\'efini, l'introduction de commandes de saut de page, de saut de ligne,
    de modification des marges . . .
\\CR.Les commandes des deux niveaux sont
  \\POL : 14.en majuscule et sous leur forme r\'eduite\\FIN : POL. ;
  aucune de ces conventions n'est une contrainte, mais elles facilitent
  la visualisation.
```

6.2 Fichier decode en format "S"

```
S0000000738C
S11B0000000000006000FFFF0001000C000C07BEFFFFFFFFFFFFFFFF0A
S11B0018FFFF02F400000003FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE5
S11B0038FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFCC
S11B0058FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB4
S11B0078FFFFFFFFFFFFFFFF002A0000000000000278000000030000C8
S11B009000000000946049E00053E7007113F40100300222202FF00815
S11B00A8FFFF00460000000000028000000E3000000000000086625
S11B00C005B9006FE7007102F405001300BC2053F054E7007102F403CD
S11B00D8001300 5E202EF058E7007102F408002200 5E2009F038FFFFE0
S11B00F000460000000000364000000E8000000000000786 05B914
S11B0108006FE7007102F406001300 5E2053F056E7007102F40B00177E
S11B012000 5E202EF054E7007102F4090022005E2009F03EFFFF002A7D
S11B013800000000000044C000000300000000000006F005B9002552
S11B0150E7007102F406002200 5E2009F030FFFF002A000000000004E
S11B016804 7C000003B000000000000065A05B900 25E7007102F40827
S11B018000 22005E2009F03CFFFF002A0000000000004B8000000 3A70
S11B0198000000000000005C405B900 25E7007102F40800 22005E2009A0
S11B01B0F03AFFFF00380000000000004F2000000 8A000000000000053
S11B01C805090 5B9004AE7007102F40000018005E202EF054E7007102 55
S11B01E0F4080022005E2009F036FFFF002A0000000000057C000008F
S11B01F800051000000000000004730 5B900 25E7007102F4FF00 22005E73
S11B02102009F052FFFF00380000000000000 5CE000000870000000D7
S11B0228000003B805B9004AE7007102F40B001E005E202EF052E700AB
S11B024071 02F407002200 5E2009F036FFFF00380000000000000656D3
S11B0258000000AD000000000000002FD05B9004AE7007102F404001371

S11B027000 5E202EF058E7007102F40000016005E2009F056FFFF003817
S11B0288000000000000704000000640000000000000242 05B9004A9F
S11B02A0E7007102F404001E005E202EF04EE7007102F401002200 5E19
S11B02B82009F016FFFF00380000000000000768000000 5B000000000FB
S11B02D000003187 05B9004AE7007102F40500110 0BC202EF054E700E9
S11B02E871 4000022005E2009F008FFFF0000205052454641 4345DE
S11B0300204C 2062752420646520636520726170706F72742 06573B4
S11B03187420646520666 16D696C6961726973657220756E207 57469DF
S11B03306C697361746575722061766563206C27656D706C6F692064CC
S11B03487520666F726D617465757220224654544C222E204365207209
S11B0360617070 6F72742061201274122072126469671220102042705
S11B0378456 36F6C65204612641272616C6520506F6C797465636 86E19
S11B03906971756520646520 4C617573616E6E6520102 06C2761696 44C
S11B03A8652064656C276F7574696C207175276 96C20641263726974E1
S11B03C02E20496C207265707265 6E6420636573206469666612726511
S11B03D86E747320706F696E7473203A4C652070726 56D6965722070 48
S11B03F06F696E742070721273656E746520756E6520646573637269 02
S11B04087074696F6E20647520666F726D617465757220617665320E1
S11B042073657320636F6D6D616E6465732038 20696C207265707265 10
S11B04386E646C61206C6F67697175652064752066 6F726D6174616789
S11B045065206574206412666696E692070617220 6C6120737569746 55C
S11B04682075 6E6520737 96E7461786520646573206 36F6D6D616E6489
S11B048065732038 20696C20641266696E691267616C656D656E74207D
S11B0498746F7573206C657320746572 6D657320746563686E697175E8
S11B04B065732065 6D706C6F791273206466 16E7320636520 6D616E759E
S11B04C8656C2E204C6520646575869136D6520706F696E7420702D8
S11B04E01273656E7465206C6520666F726D617420227374616E646178
S11B04F87264222E4C652074726F697369136D6520706F696E7420709 8
S11B05107212736 56E74652 06C657320636F6D6D616E646573206475F8
S11B0528206E6976656175206 76C6F62616C2E204C6520717561747232
S11B05406913 6D6520706F696E7420702123656E7465206C65732050
S11B0558636F6D6D616E646573206475206E6976656175206C6F636170
S11B05706C2E4C652063696E71756913 6D6520706F696E7420702123 8
S11B0588736 56E7465206C27656E73656D626C652064657320636F6D7F
S11B05A06D616E646573206 36C617373126573207061722 06F72647 6D
S11B05B8652061 6C7068616212746971756561766563207 06F7572205B
S11B05D0636861717565206 36F6D6D616E6465207361207 3796E7461F1
S11B05E8786520657420736120731 6D616E746971756 2E4C652073B2
S11B0600697869136D6520706F696E7420702123656E7465206C6541
S11B0618732064696666127265 6E74732074797065732064652070 6F1F
S11B06306C6963652065742064652066 6F6E7465206469737 06F6E69DD
S11B0648626C65732E204C652073657074691 36D6520706F696E74205D
S11B0660702123656E7652075 6E652 06C69737465206427657272CE
S11B0678657572732064127465637412657320706172206C6520666F2E
S11B0690726D617465757220 6C6F72732064652 06C27696E74726F6442
S11B06A87563746 96F6E20646573206 36F6D6D616E6465732064616E1E
S11B06C073206C652074657874652010206666F726D617465722E4C65E1
S11B06D820687569746913 6D6520706F696E7420702123656E746561
S11B06F020646575782656D706C65732064277574696C697361EF
S11B0708746 96F6E20646573206 36F6D6D616E6465732064752066 6FFA
S11B0720726D617465757220 3B206C657320636F6D6D616E6465732007
S11B07386E652073 6F6E7420706173207467573207574696C697F
S11B0750731265732038 206E6F7573207072123656E746F6E732073 3F
S11B0768657 56C656D656E74206C657320706C7573206 36F7572616E26
S11B0780746573 2E4C65206E6575766913 6D6520706F696E7420702 8A
S11B0798127365 6E7465207 56E652064657363726970746 96F6E206464
S11B07B0752073797374136D6520646520 6D6163726F732061696E7387
S11B07C869207175652 06C6575726D1274686F6465206427757 4696C6D
S11B07E0697361746 96F6E2E5 06F75722064657320726973 6F6E7387
S11B07F820646520746 56D70732C206C652070726 56D6965722070 6F83
S11B0810696E742 06E2761207075201574726520721264696712201 0CC
S11B08282 06C27616964652064752 0666F726D617465757222465470
S11B08404C222E20FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4
S11B0858FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9C
S11B0870FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF84
S11B0888FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF6C
S11B08A0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF54
S11B08B8FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3C
S11B08D0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF24
S11B08E8FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0C
S11B0900FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3
S11B0918FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFDB
```

6.3 Fichier encodé en format "SSM" (à titre d'exemple)

```
Dumpfile of MATHIEU:JEU.SM     Binary file     load address: 0000     length: 4400     start address: 0000

0000  FF FF 00 22 00 07 FF FF 00 0C 00 0C 07 BE FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 03 FF FF
0020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0080  00 00 00 01 00 04 00 00 00 00 00 00 05 00 02 00 00 00 00 00 00 07 00 07 00 00 00 00 00 0E 00 06 00 00
00A0  00 00 00 14 00 03 00 00 00 00 00 00 17 00 07 00 00 00 00 00 00 1E 00 03 00 00 FF FF FF FF FF FF FF FF
00C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0140  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0160  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0200  00 2A 00 00 00 00 00 01 BA 00 00 00 15 00 00 00 00 00 00 00 09 26 05 B9 00 14 E7 00 71 01 F4 02
0220  00 12 01 E1 20 00 F0 16 FF FF 00 2A 00 00 00 00 00 00 01 A0 00 00 1F 00 00 00 00 00 00 08 E1
0240  05 B9 00 16 E7 00 71 02 F4 06 00 14 01 82 20 00 F0 20 FF FF 00 2A 00 00 00 00 00 00 01 C0 00 00
0260  00 1C 00 00 00 00 00 00 08 9C 05 B9 00 16 E7 00 71 02 F4 06 00 14 01 B0 20 00 F0 1C FF FF 01 0A
0280  00 00 00 00 00 00 01 DC 00 00 00 04 AC 00 00 00 00 00 5E 04 B9 04 FC 03 B4 E7 00 71 02 F4 01 00 19
02A0  00 5E 23 95 F0 46 E7 00 71 02 F4 08 00 1A 00 5E 23 63 F0 44 E7 00 71 02 F4 04 00 18 00 5E 23 28
02C0  F0 46 E7 00 71 02 F4 09 00 16 00 5E 22 ED F0 4A E7 00 71 02 F4 07 00 1A 00 5E 22 B2 F0 4E E7 00
02E0  71 02 F4 08 00 1A 00 5E 22 77 F0 46 E7 00 71 02 F4 0A 00 1C 00 5E 22 3C F0 44 E7 00 71 02 F4 0D
0300  00 1A 00 5E 22 01 F0 46 E7 00 71 02 F4 00 00 16 00 5E 21 C6 F0 4E E7 00 71 02 F4 09 00 1A 00 5E
0320  21 88 F0 46 E7 00 71 02 F4 03 00 1A 00 5E 21 50 F0 4A E7 00 71 02 F4 0A 00 1A 00 5E 21 15 F0 48
0340  E7 00 71 02 F4 0A 00 1A 00 5E 20 DA F0 48 E7 00 71 02 F4 02 00 19 00 5E 20 A8 F0 48 E7 00 71 02
0360  F4 0C 00 1A 00 5E 20 6D F0 46 E7 00 71 02 F4 07 00 16 00 5E 20 32 F0 4A E7 00 71 02 F4 06 00 14
0380  00 5E 20 00 F0 38 FF FF 00 00 20 45 58 45 4D 50 4C 45 20 44 27 55 54 49 4C 49 53 41 54 49 4F 4E
03A0  20 4D 61 74 68 69 65 75 20 2D 20 53 63 68 6D 69 74 7A 20 26 20 44 61 76 69 64 20 4C 65 76 79 20
03C0  20 4C 61 75 73 61 6E 6E 65 2C 20 31 32 20 2D 20 61 76 72 69 6C 20 2D 20 31 39 38 33 20 4C 65 20
03E0  74 65 78 74 65 20 73 6F 75 72 63 65 20 73 65 20 63 6F 6D 70 6F 73 65 20 64 65 20 63 61 72 61 63
0400  74 13 72 65 73 20 63 6F 6E 73 74 69 74 75 61 6E 74 20 6C 65 73 20 63 6F 6D 6D 61 6E 64 65 73 20
0420  64 75 6E 69 76 65 61 75 20 67 6C 6F 62 61 6C 2C 20 64 75 20 6E 69 76 65 61 75 20 6C 6F 63 61 6C
0440  20 65 74 20 6C 65 20 63 6F 6E 74 65 6E 75 2E 20 6C 65 20 63 6F 6E 74 65 6E 75 20 6E 27 61 20 61
0460  75 63 75 6E 65 20 69 6D 70 6F 72 74 61 6E 63 65 2E 20 4C 65 73 20 63 6F 6D 6D 61 6E 64 65 73 20
0480  64 75 20 6E 69 76 65 61 75 20 67 6C 6F 62 61 6C 20 73 6F 6E 74 20 6C 65 73 20 63 6F 6D 6D 61
04A0  6E 64 65 73 20 73 69 74 75 12 65 73 61 76 61 6E 74 20 6C 61 20 73 12 71 75 65 6E 63 65 20 5C 64
04C0  6F 63 75 2E 20 43 65 73 20 63 6F 6D 6D 61 6E 64 65 73 20 6F 6E 74 20 12 74 12 20 63 68 6F 69 73
04E0  69 65 73 20 61 66 69 6E 20 64 65 20 70 72 12 73 65 6E 74 65 72 20 6C 65 73 20 64 69 66 66 12 72
0500  65 6E 74 65 73 20 70 6F 73 73 69 62 69 6C 69 74 12 73 20 64 65 20 64 65 73 69 6E 69 74 69 6F 6E
0520  73 20 64 27 69 6E 64 69 63 61 74 69 6F 6E 73 20 64 65 20 66 6F 72 6D 61 74 61 67 65 20 65 74 20
0540  6E 6F 6E 20 70 6F 75 72 20 6F 62 74 65 6E 69 72 20 75 6E 65 20 70 72 12 73 65 6E 74 61 74 69 6F
0560  6E 20 63 6F 6E 76 65 6E 61 62 6C 65 2E 20 4C 61 20 6C 69 73 74 65 20 6E 27 65 73 74 20 70 61 73
0580  20 6E 27 65 73 74 20 70 61 73 65 78 68 61 75 73 74 69 76 65 20 3B 20 70 61 72 20 65 78 65 6D 70
05A0  6C 65 2C 20 75 6E 20 6E 75 6D 12 72 6F 74 61 74 61 67 65 20 64 65 73 20 70 61 67 65 73 20 65 6E
05C0  6E 75 6D 12 72 6F 74 61 67 65 20 65 6E 20 68 61 75 74 20 64 65 20 70 61 67 65 2C 20 75 6E 65 20
05E0  74 61 62 6C 65 20 64 65 73 20 6D 61 74 69 13 72 65 73 20 10 20 74 72 6F 69 73 20 6E 69 76 65 61
0600  75 78 2C 20 6C 61 20 66 6F 6E 74 65 20 64 65 20 74 79 70 65 69 74 61 6C 69 71 75 65 2C 20 6C 61
0620  20 70 6F 6C 69 63 65 20 6E 75 6D 12 72 6F 20 31 31 20 65 74 20 6C 27 69 6E 74 65 72 6C 69 67 6E
0640  65 20 10 20 33 20 6D 69 6C 6C 69 6D 13 74 72 65 73 20 70 6F 75 72 20 6C 65 20 72 12 73 75 6D 12
0660  2C 20 6C 65 73 20 12 6C 12 6D 65 6E 74 73 20 64 27 69 64 65 6E 74 69 66 69 63 61 74 69 6F 6E 73
0680  20 63 65 6E 74 72 12 73 20 73 75 72 20 6C 61 20 70 72 65 6D 69 13 72 65 20 70 61 67 65 20 28 70
06A0  61 67 65 20 6E 6F 6E 20 6E 75 6D 12 72 6F 74 12 65 29 20 2E 2E 2E 4C 65 73 20 63 6F 6D 6D 61 6E
06C0  64 65 73 20 64 75 20 6E 69 76 65 61 75 20 6C 6F 63 61 6C 20 6D 6F 64 69 66 69 65 72 6F 6E 74 20
06E0  6C 27 12 6C 12 6D 65 6E 74 20 6C 6F 67 69 71 75 65 20 63 6F 75 72 61 6E 74 20 38 20 70 61 72 20
0700  65 78 65 6D 70 6C 65 2C 20 73 70 12 63 69 66 69 63 61 74 69 6F 6E 20 64 27 75 6E 65 20 66 6F 6E
0720  74 65 2C 20 64 27 75 6E 65 20 70 6F 6C 69 63 65 2C 20 65 74 20 64 27 75 6E 20 73 6F 75 6C 69 67
0740  6E 65 6D 65 6E 74 20 64 69 66 66 12 72 65 6E 74 20 64 65 20 63 65 6C 75 69 20 64 12 66 69 6E 69
0760  " 20 69 6E 74 72 6F 64 75 63 74 69 6F 6E 20 64 65 20 73 61 75 74 20 64 65 20 70 61 67 65 2C 20
0780  6F 65 73 61 75 74 20 64 65 20 6C 69 67 6E 65 2C 20 6D 6F 64 69 66 69 63 61 74 69 6F 6E 20 64 65
07A0  73 20 6D 61 72 67 65 73 2E 2E 2E 20 4C 65 73 20 63 6F 6D 6D 61 6E 64 65 73 20 64 65 73 20 64 65
07C0  75 78 20 6E 69 76 65 61 75 78 73 6F 6E 74 20 63 68 61 71 75 65 20 66 6F 69 73 20 70 6C 61 63 12
07E0  65 73 20 65 6E 20 64 12 62 75 74 20 64 65 20 6C 69 67 6E 65 2C 20 65 6E 20 6D 61 6A 75 73 63 75
0800  6C 65 20 65 74 20 64 61 6E 73 20 6C 65 75 72 20 66 6F 72 6D 65 20 72 12 64 75 69 74 65 20 38 20
0820  61 75 63 75 6E 65 20 64 65 20 63 65 73 20 63 6F 6E 76 65 6E 74 69 6F 6E 73 20 6E 27 65 73 74 20
0840  75 6E 65 20 63 6F 6E 74 72 61 69 6E 74 65 20 20 6D 61 69 73 20 65 6C 6C 65 73 70 65 72 6D 65 74
0860  74 65 6E 74 20 64 65 20 66 61 63 69 6C 69 74 65 72 20 6C 61 20 76 69 73 75 61 6C 69 73 61 74 69
0880  6F 6E 20 64 65 73 20 63 6F 6D 6D 61 6E 64 65 73 2E 20 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
08A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
08C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
08E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0900  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0920  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0940  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0960  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0980  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
09A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
09C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
09E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0A00  00 2E 00 00 00 00 00 00 00 AE 00 00 00 10 00 00 00 00 00 00 09 26 05 B9 00 14 E7 00 71 01 F4 02
0A20  00 12 02 13 20 00 E5 82 F0 10 E5 02 FF FF 00 7E 00 00 00 09 00 00 00 BE 00 00 02 03 00 00 00 00
0A40  00 00 07 88 05 B9 01 6F E7 00 71 02 F4 00 00 12 00 5E 21 50 F0 52 E7 00 71 02 F4 06 00 15 00 00
0A60  21 1E F0 58 E7 00 71 02 F4 08 00 16 00 00 20 E3 F0 52 E7 00 71 02 F4 0A 00 17 00 00 20 B1 F0 56
0A80  E7 00 71 02 F4 01 00 18 00 00 20 76 F0 54 E7 00 71 02 F4 0D 00 1A 00 00 20 38 F0 50 E7 00 71 02
```

```
--"----------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
---g-------1-------(-------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
-*----------1------ &---@--q---
-@-- --1---*----------)---------
---!--q----@-- -- ---*----------
-0------------!-?q----@-- --@----
----------------^--------q----Q
-^"H--F-?q----Q-f"Hc-D--q----b-^"H(
-F--q----(-^"--U--q----b-^"--N--
q----Q-^"w-F--q----b-^"(-D--q---
-Q-^"--F--q----!-^!--N-?q-- -Q-^
!--F--q----Q-^!P-J--q----Q-^!0-H
--q----b-^ --H-?q----Q-^ --H--q-
---Q-^ @-F--q----!-^ 2-J--q----@
-^ --8---- EXEMPLE D'UTILISATION
 Mathieu - Schmitz & David Levy
 Lausanne, 12 -avril -(1983 Le
texte source se;compose;de caract
ères constituant les commandes
 duniveau global; du niveau local
 et le contenu.;Le contenu n'a a
ucune importance. Les commandes
du niveau global sont les comma
ndes situées;avant la séquence \d
ocu. Ces commandes ont été chois
ies afin de présenter les différ
entes possibilités de définition
s d'indications;de formatage et
non pour obtenir une présentatio
n convenable du texte. La liste
 n'est pas;exhaustive ; par exemp
le, un numérotage des;pages -
numérotage en haut de page, une
table des matières à trois nivea
ux, la fonte de;typeitalique, la
 pol'-e numéro 11 et l'interlign
e à 3 millimètres pour le résumé
, les éléments d'identifications
 centrés sur la première page (p
age non numérotée) ...Les comman
des du niveau local modifieront
 l'élément logique courant ; par
 exemple; spécification d'une fon
te, d'une police, et d'un souslig
nement différent de celui défini
, introduction de saut de page,
 desaut de ligne; modification de
s marges... Les commandes des de
ux niveaux;sont chaque fois placé
es en début de ligne, en majuscu
le et dans leur forme réduite ;
aucune de ces conventions n'est
une contrainte mais elle;permet
tent de faciliter la visualisati
on des commandes. ----------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
----------------------------------
-.----------@------ &--p@--q---
-@-@ --+@-----"----------------
-------@--q----@-^!P-R--q----@--
!@-X--q----!-- +R--q----!-- --V
--q----@-- v-T--q----U-- ;-P--q-
```

7. COMMANDES D'UTILISATION

Pour exécuter le programme de décodage "BINAIRES", l'utilisateur rentrera la commande :

```
@DECODE <FICHIER 1> <FICHIER 2>
```

où

<FICHIER 1> est le fichier d'entrée en format "BINAIRE" et

<FICHIER 2> le fichier en sortie en format "S"

Pour exécuter le programme de décodage "SBINAIRE", l'utilisateur rentrera la commande :

```
@FORS <FICHIER 1> <FICHIER 2>
```

où

<FICHIER 1> est le fichier d'entrée en format "S" et

<FICHIER 2> le fichier en sortie en format "BINAIRE"

ANNEXE E

DESCRIPTION TECHNIQUE

DE

L'IMPRIMANTE LBP-10

INTRODUCTION

Cette annexe contient la description de l'imprimante laser CANON LBP10

INTRODUCING THE

Canon LASER BEAM PRINTER

LBP-10

Prepared by
NEW ENTERPRISES DIVISION

CANON INC
3-11-28 Mita, Minato-ku, Tokyo, Japan

CANON LBP-10 Revision-1 MAR 1980 PRINTED IN JAPAN

CONTENTS

1. INTRODUCTION

The LBP-10 is a non-impact, high-speed, general-purpose printer in which an electrophotographic process, laser optics and electronic control circuits are combined. Because a laser beam is employed, the LBP-10 is capable of printing high quality characters and graphics on plain paper very quietly and very quickly. The LBP-10 is intended for use with small business computers, minicomputers, word processors, and other on-line/off-line processor systems.
The LBP-10's versatility will be most appreciated in such sophisticated applications as data processing, word processing, graphics, facsimile, electronic mail, etc.
This document gives basic information required for the reader to understand the LASER BEAM PRINTER LBP-10.

2. PRINTING METHOD

The LBP-10 is based on Canon's own NP-electrophotographic technology and precision laser optical technology.
The LBP-10 uses the Canon Semiconductor Laser as a light beam generator.
The laser beam of the Canon Semiconductor Laser is modulated by an electrical signal sent from the printer control unit. The beam is then deflected by a polygonal mirror and focused onto the surface of the NP-photosensitive drum, where it forms an electrostatic latent image of the data to be printed. This latent image is then developed and transferred onto a sheet of plain paper.

3. SPECIFICATIONS

3.1 GENERAL

Type:	Desktop type page printer
Print Method:	NP-Electrophotography + Laser beam scanning
Print Speed:	10 letter-size sheets per minute (Time for first page —— 13 ~ 96 sec.)

3.2 OPTICAL

Laser Beam Generator:	Canon Semiconductor Laser
Scanning System:	Polygonal mirror scanner
Dot Pitch:	Horizontal —— Variable with horizontal clock Vertical —— 240 raster lines/inch

3.3 PROCESS

Process Speed:	59.7 mm/sec.
Photosensitive Drum:	3-layer construction (conductive metal, photo-conductive layer, insulating layer on top)
Development Method:	Liquid-Dry (Canon NP Method, wet toner with automatic feed to heat fixing).

3.4 PRINT PAPER

Paper Type:	NP standard plain paper (cut sheets)
Paper Weight:	64 ~ 80 g/m²
Paper Loading:	Cassette method

Cassette Capacity:	190 sheets (75 g/m²) per cassette NOTE: Paper can be piled up to 17 mm.		
Cassette Types & Paper Sizes Available:	A4	210 x 297 mm	8.3 x 11.7 inch
	Letter	216 x 279 mm	8.5 x 11 inch
	G. Letter	203 x 267 mm	8 x 10 inch
	Legal	216 x 356 mm	8.5 x 14 inch
	G. Legal	203 x 330 mm	8 x 13 inch
	Foolscap	216 x 330 mm	8.5 x 13 inch
	A. Foolscap	206 x 337 mm	8.1 x 13.3 inch
	Folio	210 x 330 mm	8.3 x 13 inch
	Statement	140 x 216 mm	5.5 x 8.5 inch

3.5 ELECTRICAL

Power Requirements:	~~120~~ 100 V AC ± 10%, ~~60~~ 50 Hz ± 1 Hz
Power Consumption:	0.96 KVA (8 Amps. max. 120 VAC)

3.6 ENVIRONMENTAL

Temperature:	Operating	10 ~ 35°C
	Non-Operating	0 ~ 35°C
Humidity:	Operating	20 ~ 80% RH
	Non-Operating	20 ~ 80% RH
Warm up Time:	14 ~ 75 sec. (Depending on environmental conditions.)	
Noise Level:	Under 57 dB(A) (Except intermittent impact noise)	

3.7 PHYSICAL

Dimensions:	874 mm (W) x 443 mm (H) x 538 mm (D)
Weight:	95 kg

4. LASER RADIATION SAFETY INFORMATION

4.1 LASER SAFETY

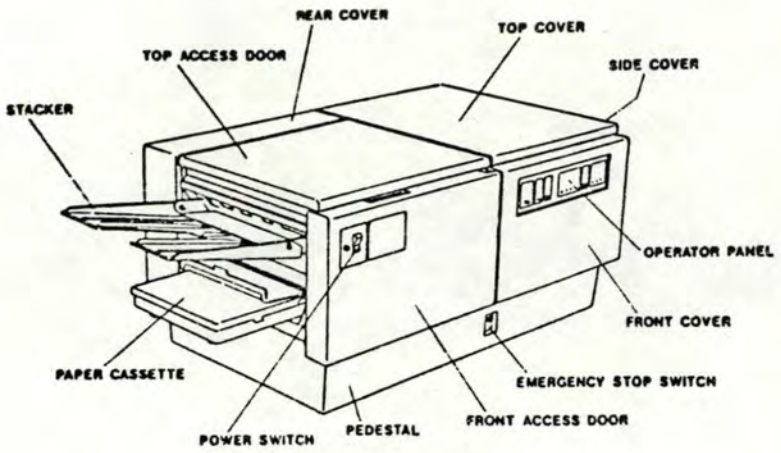
The LBP-10 is certified to comply with DHEW Radiation Performance Standards under the Radiat
Control for Health and Safety Act of 1968 as a Class 1 Laser Product.
This means that the LBP-10 belongs to a class of laser products that does not produce hazardous la
radiation.
Since the radiation emitted inside the LBP-10 is completely confined within protective housings a
external covers, the laser beam cannot escape from the machine during any phase of user operati

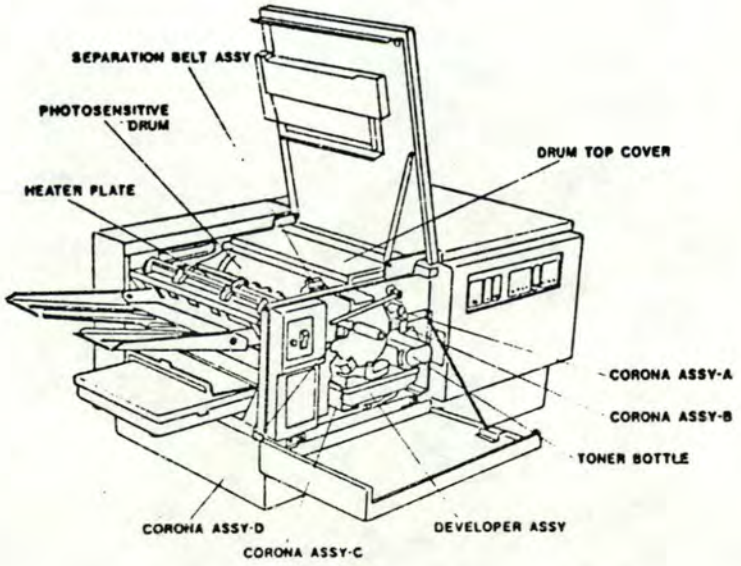
4.2 BRH REGULATIONS

Effective August 2, 1976, the Bureau of Radiological Health (BRH) of the Food and Drug Administration has established regulations for laser products.
The regulations apply to laser products manufactured after August 1, 1976.
To the right is the PRODUCT IDENTIFICATION AND CERTIFICATION LABEL required by the BRH regulations.

CANON INC. AA3 0014
3-30-2 SHIMOMARUKO. OHTA-KU. TOKYO. JAPAN
CANON U.S.A., INC.
10 NEVADA DRIVE. LAKE SUCCESS. LONG ISLAND
N.Y. 11042. U.S.A.
MANUFACTURED:
SERIAL NUMBER:
THIS PRODUCT CONFORMS WITH DHEW RADIATION PER
FORMANCE STANDARD 21CFR CHAPTER I SUBCHAPTER J

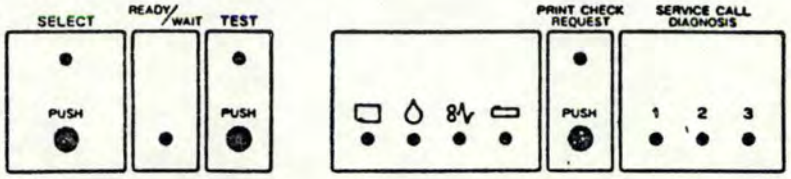
5. EXTERNAL & INTERNAL VIEW





6. SWITCHES & INDICATORS

6.1 OPERATOR PANEL



(1) SELECT switch & indicator

When the power switch is turned ON, the LBP-10 is in DE-SELECT MODE for local operation, regardless of interface signals. When SELECT switch is pressed, SELECT MODE is specified, and the LBP-10 is controlled by external equipment through the interface signals.

(2) READY/WAIT indicator

This indicator flashes during warming up operation, and is lit continuously when the LBP-10 is ready to accept a print command.

(3) TEST switch & indicator

This switch is used to check the function of the LBP-10 internally and is only effective in DE-SELECT MODE.
When the TEST switch is pressed, the LBP-10 prints out a prestored pattern continuously.
When the TEST switch is pressed again, the LBP-10 terminates printing.

(4) ▭ indicator

This indicates that the paper cassette is empty.

(5) 💧 indicator

This indicates a low developer liquid level.

(6) ⚡ indicator

This indicates a paper jam.

(7) ▭ indicator

This indicates that the toner bottle is empty.

(8) PRINT CHECK REQUEST indicator & RESTART switch

This indicates that there is a possibility of existence of print defects in the last page printed out. Therefore the operator is requested to examine and take out that copy from the stacker. After this procedure the operator must push the Restart switch to restart printing. Then the outside equipment initiates transmission of the data of that page as requested by a Data Retransfer signal.

(9) SERVICE CALL DIAGNOSIS indicators

These indicators are provided for quick and appropriate repair service.
Each of the three LEDs has its own I-D number, and indicates that a failure has occurred in a specified module, requiring a trained technician.
Therefore, the operator is asked to request repair service and to inform the service receptionist of the I-D numbers of the flashing LEDs.

6.2 POWER switch & indicator

This switch turns off AC power supplied to all components of the LBP-10 except the environment conditioning circuits.

6.3 EMERGENCY STOP switch

In the event of unexpected accidents, such as fire, smoke, spark, etc, an EMERGENCY STOP switch is provided on the front side of the pedestal.
This immediately cuts the AC power supply regardless of the POWER switch. Therefore all electric power inside the LBP-10, including environment conditioning circuits, is turned off with this switch. This switch must be kept normally on, and should be turned off only in an emergency.

6.4 COUNTER (6-digits, non-resettable)

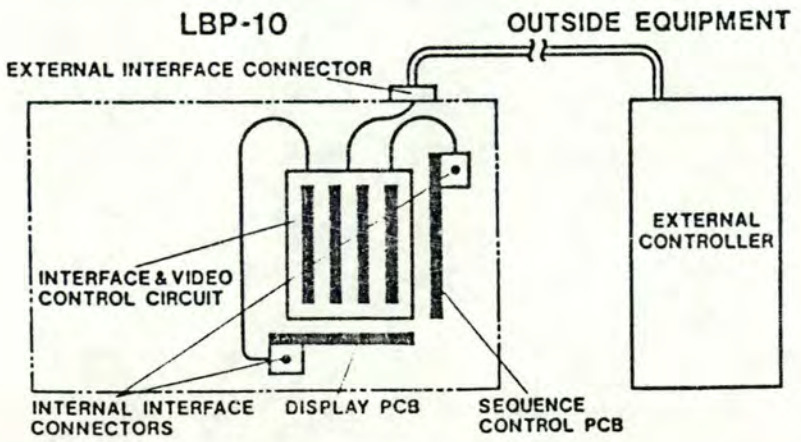
This indicates the total number of print-outs made by the LBP-10.

7. SYSTEM INTERFACE

This section explains the types of interfaces available and the rough sequence of the principal interface signals.
For detailed interfacing procedure, you are suggested to refer to the LBP-10 System Interface Manual.

7.1 INTERFACE TYPES

The LBP-10 can be interfaced either at sequential print video signal level or at coded signal level such as ASCII and EBCDIC Codes.
The standard LBP-10 is equipped with a standard video signal interface. A coded signal interface or other special video signal interfaces are optional.
Standard video signal interfacing is realized through the Sequence Control P.C.B (SQC) provided in every LBP-10, while coded signal interfacing requires an optional Interface & Video Control Circuit (I.V. Circuit) which varies in form depending on the application.
The names of the components related to interfacing are defined as follows.



NOTES:

1. Interface & Video Control Circuit (I. V. Circuit) is an optional unit for the page printer.
2. External Interface Connector is used to connect between LBP-10 and outside equipment.
3. Two Internal Interface Connectors are used, one to connect S.Q.C. and I. V. Circuit, and the other to connect Display P.C.B. and I. V. Circuit.

7.2 INTERFACE SEQUENCE

When the power switch is turned on, power is supplied to all parts of the LBP-10 and the warming up sequence is initiated. On completion of this sequence, I. V. Circuit (Interface & Video Control Circuit) is notified of ready status by means of I-READY signal from the S.Q.C (Sequence Control P.C.B.). The I. V. Circuit controls the data flow between the printing engine and the outside equipment. The I. V. Circuit covers such functions as interfacing the LBP-10 with various equipment, font generation, page buffering, form overlay, etc.
The I. V. Circuit receives signals from outside equipment and outputs video print signals to the laser modulator driver.
At this moment, S.Q.C has been transmitting horizontal sync. signals, I-BD, which are a series of periodic pulses. Then I. V. Circuit transmits I-PRINT to S.Q.C. to start rotation of the photosensitive drum, and S.Q.C. transmits a vertical sync. signal, I-TOP, to I. V. Circuit. Upon receipt of I-TOP, I. V. Circuit stops sending I-PRINT.
Syncronizing vertically with I-TOP and also horizontally with I-BD, I. V. Circuit transmits an image modulated video signal I-VIDEO.
When the printing cycle for one page is finished, S.Q.C. transmits I-PRINT END to I. V. Circuit.
For continuous printing, I. V. Circuit checks I-READY again and repeats the sequence.
When the LBP-10 is not ready for printing, the reason can be learned by checking I-STATUS signal.
The vertical scanning pitch is fixed and is 240 lines/inch (9.45 lines/mm), while the horizontal dot pitch is varied depending on the video frequency.
When the video frequency is 1.84 MHz, the horizontal dot spacing is equal to the vertical scanning pitch, i. e. 240 dots/inch.

8. PAPER SIZE & THROUGHPUT

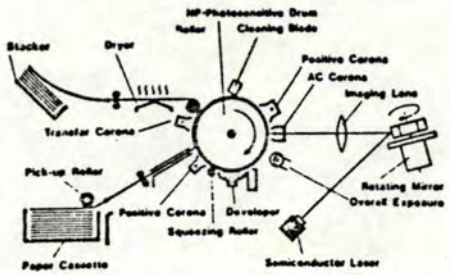
As shown in the specifications nine kinds of printer paper are available. Separate paper cassettes are provided for each size of paper.
When a paper cassette is set in the LBP-10, the corresponding paper size signal is transmitted to I. V. Circuit in a form of the I-STATUS signal. The paper size signal may be used to obtain the maximum throughput in printing.

9. PRINTING SYSTEM

9.1 PRINTER SCHEMATIC

The video signal generated in I. V. Circuit is transmitted to the laser driver in the printing engine. A schematic drawing of the engine is shown below.

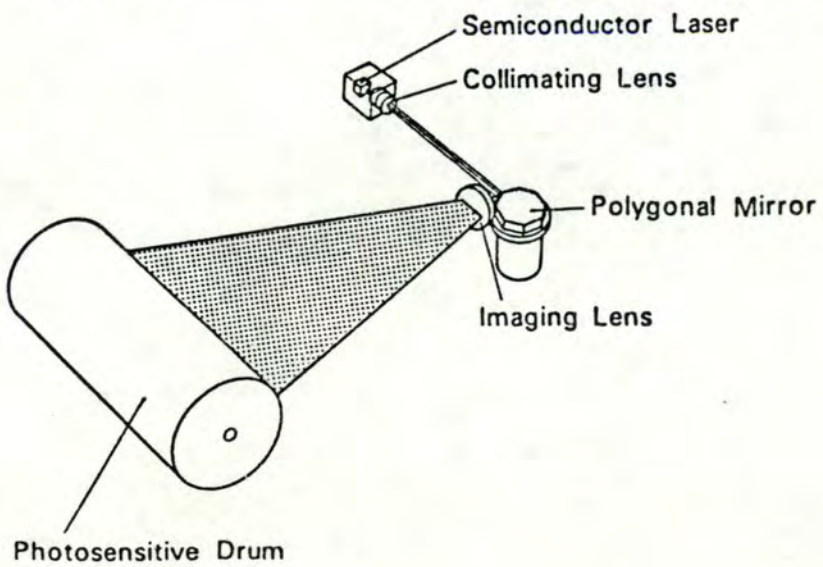
Printing Process Schematic



The engine is made up of two main units, namely, an optical unit and an electrophotographic unit.

9.2 OPTICS

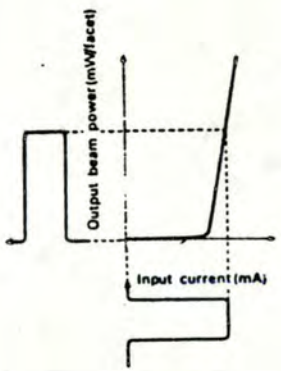
The optical unit is composed of two principal modules, i.e. a semiconductor laser module and a scanner imaging lens module.



A video signal of about 2 MHz band width sent from I. V. Circuit is led into the laser driver. The LBP-10 employs a GaAlAs double hetero-structure semiconductor diode laser as the light source.

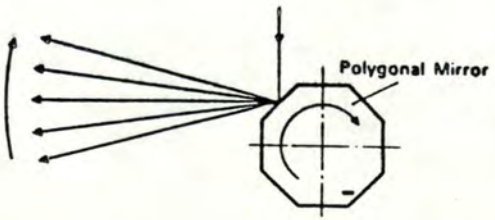
In the semiconductor laser, when injection current is increased, laser oscillation occurs instantaneously at a certain current threshold level.

The laser beam is modulated by controlling the injection current.
The following figure shows, as an example, the laser power output which is digitally modulated by the input current.



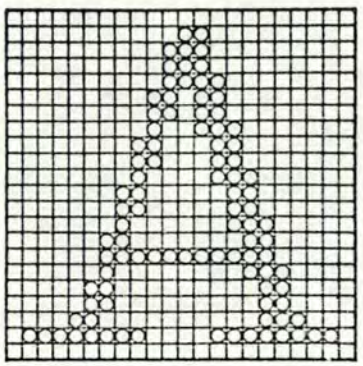
The laser is placed on the optical axis in the focal plane of the collimating lens to produce a collimated beam.

The modulated laser beam enters a high-precision optical scanner which is made of a rotating polygonal mirror. This polygon has eight facets processed as a single unit. It deflects the beam and scans the NP photosensitive drum in the rotary axial direction.



The polygonal mirror rotates at a constant angular speed so that the laser beam is deflected at a constant angular velocity.

In this system, the characters, as with television, are formed by raster scanning. The characters are designed by appropriate alignment of circular beam spots as shown in the drawing.

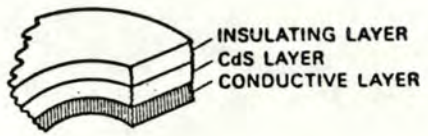


The pitch of the scanning line is about 106 microns. The manufacturing tolerance of the rotating mirror directly affects the quality of the printed pictures. For instance, the pyramidal error (face-to-axis angular tolerance), the divisional error (face-to-face angular tolerance), flatness and the revolution drift appear as pitch drift and jitters. The rotating mirror, therefore, is designed and manufactured with high precision.

The light deflected in the rotary axial direction of the NP photosensitive drum by the rotating mirror is focused on the surface of the NP photosensitive drum by the imaging lens. Since the rotating mirror is revolving at a constant speed, the reflected laser beam is deflected at a constant angular velocity. The imaging lens, in addition to focusing the deflected beam, provides distortion of $1/\cos^2$ to correct the speed.

9.3 IMAGE FORMATION BY NP PROCESS

NP electrophotography was originally developed by Canon. The composition of the NP photoreceptor is shown in the illustration (right). The photoreceptor, from the bottom, consists of a conductive substrate using aluminum base, CdS photoconductive layer and the transparent polyethylene telephthalate film insulating layer. The surface of the photoreceptor is shielded with a strong insulating film and so it is highly durable electrically, mechanically and chemically. It produces high-contrast electrostatic images, which are stable in time. The photoconductor CdS has good sensitivity to the 820 nm wavelength of the semiconductor laser.



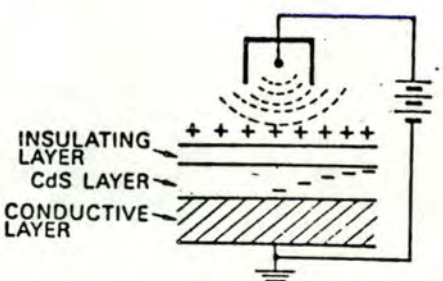
The Printing Process Schematic on page 7 shows a schematic outline of the image forming system. In general terms, the image-forming steps consists of the latent-image forming steps and the developing, transfer and fixing steps.
The drum-shaped NP photoreceptor revolves in the direction indicated by the arrow at a constant speed of 60 mm per second. The latent image forming, developing, transferring, cleaning, paper transporting, and fixing stations are located in that order around the photosensitive drum.

(1) LATENT IMAGE FORMING PROCESS

The electrostatic latent image forming process is described in the following.

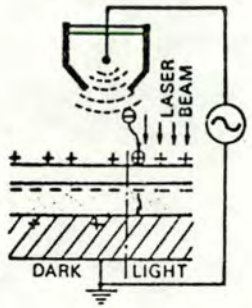
PRIMARY CHARGING

First, the surface of the insulating layer of the NP photoreceptor is uniformly charged. The charge polarity is positive when the photoconductive layer is an N-type semiconductor and negative when it is a P-type semiconductor. In case of CdS, the positive charging is selected. At this time, opposite polarity and same amount of charges as those on the surface of the insulating layer are injected into the photoconductive layer from the base substrate, thus forming a charged layer at the interface of the insulating and photoconductive layers.



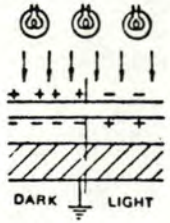
AC CORONA DISCHARGE AND SIMULTANEOUS IMAGE EXPOSURE

The second step is the AC corona discharging of the surface of the insulating layer and simultaneous emission of the modulated and deflected laser beam. The light portion on which laser beam is applied becomes conductive as the resistance of the photoconductive layer drops, and the charge on the surface and rear of the insulating layer rapidly attenuates. The potential of the dark portion, on which no laser beam is applied, becomes practically zero as the surface of the insulating layer is exposed to the AC corona discharge, while the negative charge formed at the interface of the insulating and photoconductive layers is held unchanged. As a result, a positive charge countering the negative charge is dividedly present at the insulating layer surface and the base substrate. The ratio of division is determined by the electric capacitance of the insulating and photoconductive layers. At the time in question, the surface potential is about zero at both the light and the dark portions – there being no potential contrast between the two.

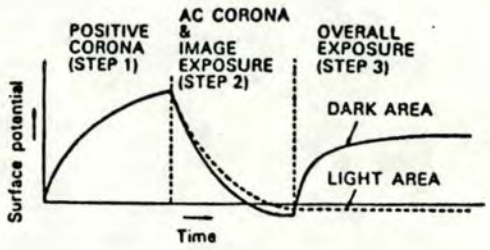


OVERALL EXPOSURE

At the third step, the entire photosensitive surface is uniformly exposed. As the light portion has already been discharged, no change takes place in the surface potential. In the dark portion, the high resistance of the photoconductive layer, abruptly drops and it assumes a state of conductivity. As a result, surplus charge at the interface of the insulating and photoconductive layers is discharged into the base substrate, leaving only a charge equal to that left on the surface of the insulating layer. Consequently, the surface potential of the insulating layer increases to the positive side. A high-contrast electrostatic latent image is thus formed. Also, as the charge is held on both sides of the insulating layer, the electrostatic latent image is stable in time and is held intact for many hours.

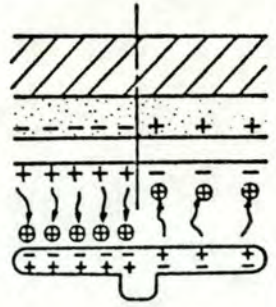


The surface potential change up to this moment is illustrated in the following.



(2) DEVELOPING, TRANSFER, FIXING AND OUTPUT

The electrostatic latent image thus formed on the drum is developed with the developer liquid. The developer liquid is made of toner particles dispersed in a solvent liquid. The positively charged toner particles are attracted to the areas of the lower potential on the drum surface on which the laser beam has been applied.



Transfer is accomplished by bringing a sheet of plain paper close to the drum surface and exposing its reverse side to the corona discharge. After the transfer, the photosensitive drum with residual toner is cleaned out to prepare for the next latent image-forming process. The NP photoreceptor is used repeatedly in this manner.

The sheets of plain paper with their transferred images are dried with the heater plate and deposited on the print-out paper stacker.

ANNEXE F

EXEMPLES

DE

FEUILLES IMPRIMEES

INTRODUCTION

Cette annexe contient quelques exemples de feuilles imprimées sur l'imprimante laser CANON LBP10 (exemple de texte et exemples graphiques).

PAGE DE DEMONSTRATION POUR IMPRIMANTE LBP10

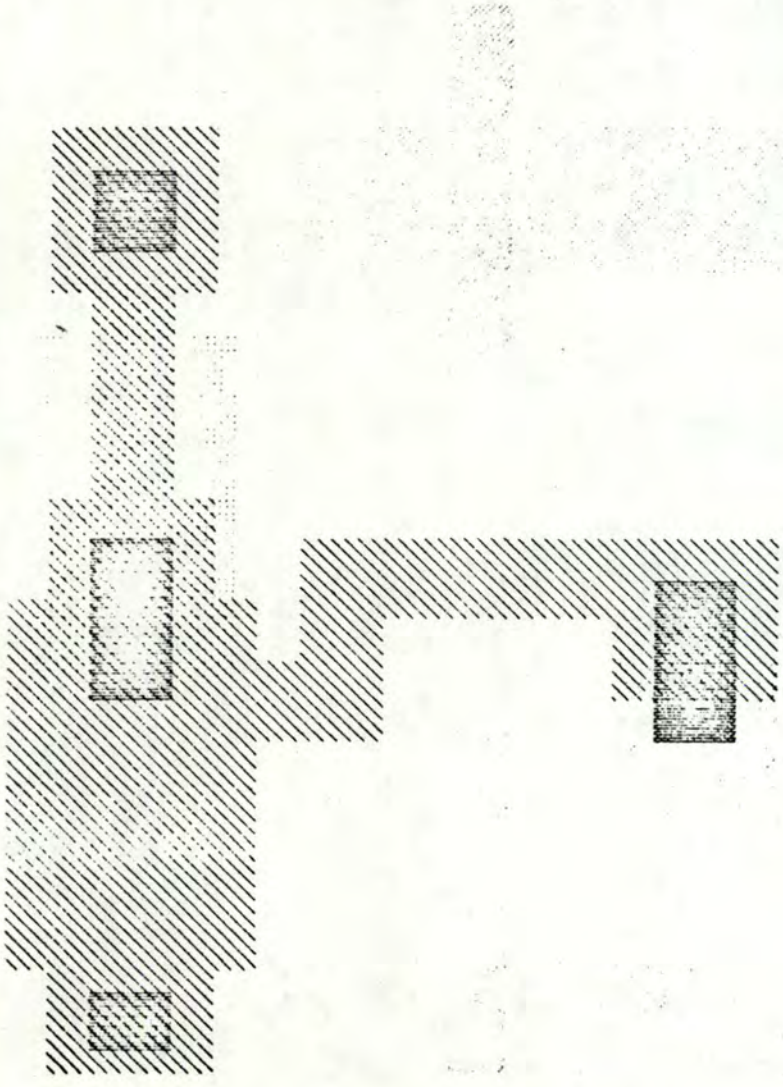
Texte et graphique mélangé...

Et se deplacer dans la page et changer de fonte...

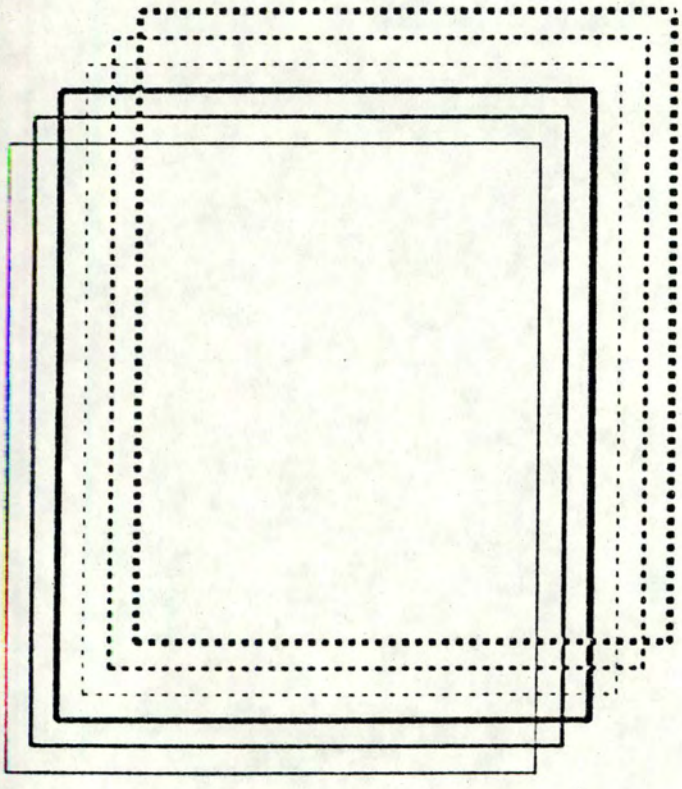
3.4 THE SHIFT REGISTER

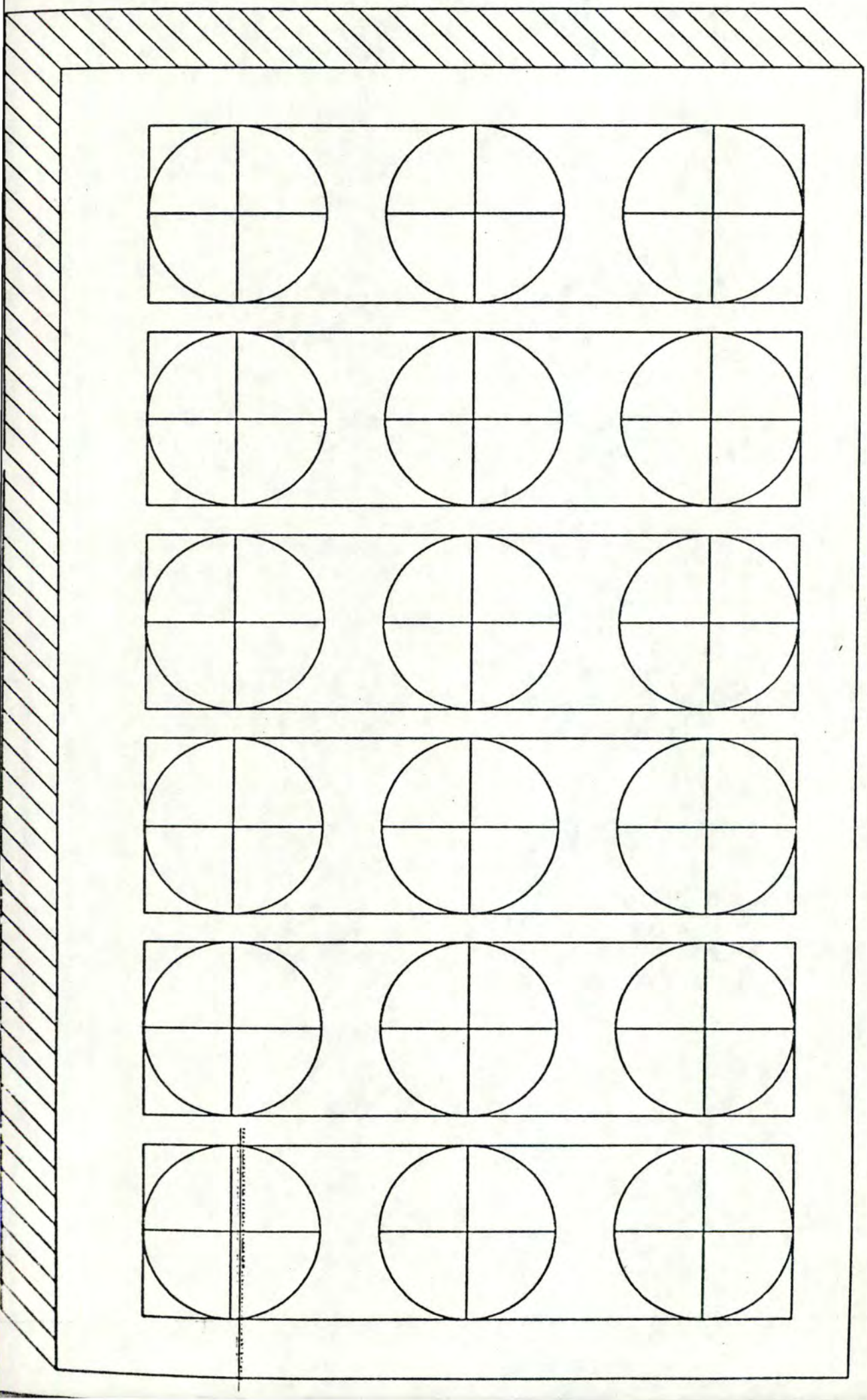
Perhaps the simplest structure for enabling the mouvement of a sequence of data bits is the serial shift register, shown in the form of a circuit diagram in Fig.3.4(a). The shift register is composed of level-restoring inverters coupled by pass transistors, with the mouvement of data controlled by applying clock signals P1 and P2 to the gates of alternate pass transistors in the sequence.

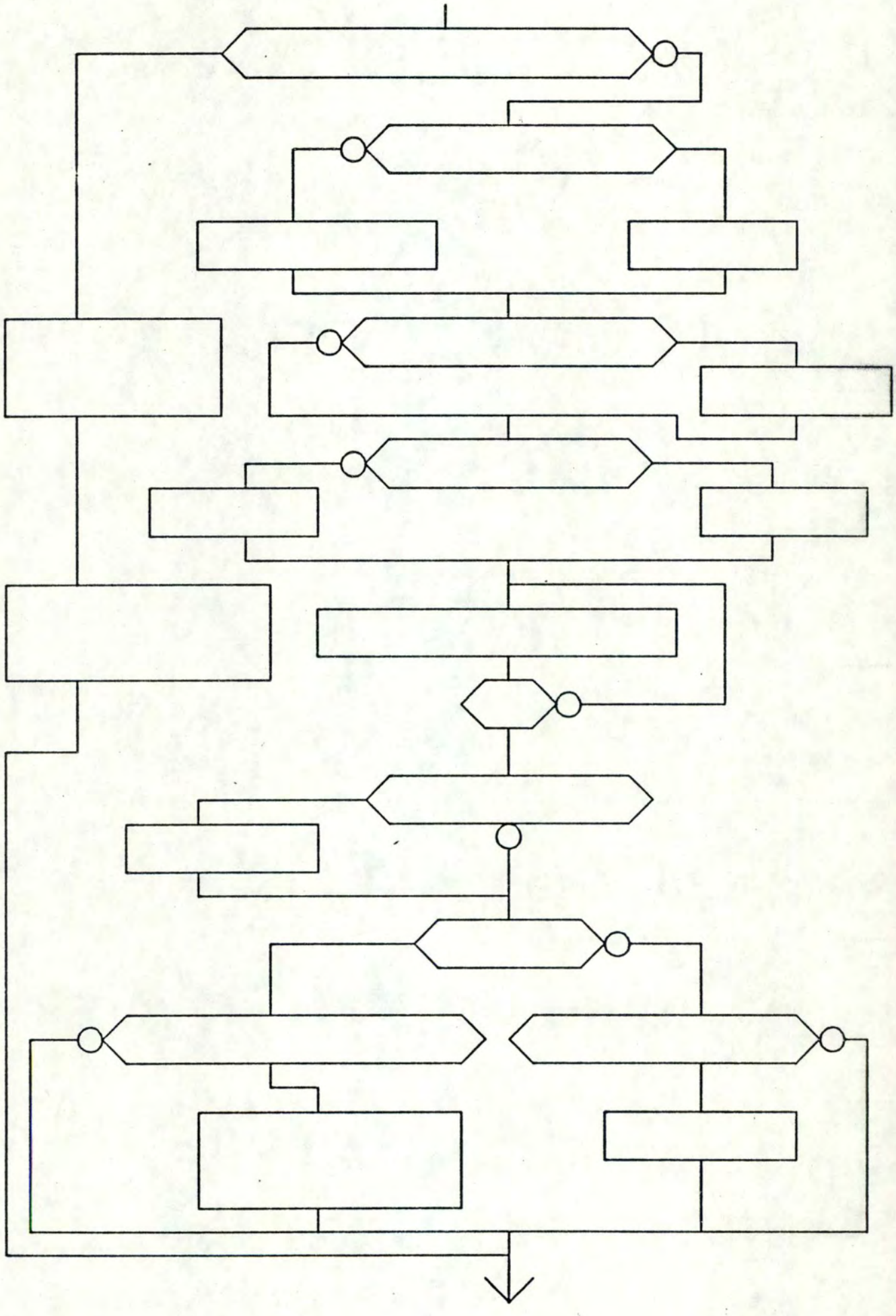
Data is shifted from left to right as follows. Suppose a logic signal X present on the leftmost input to the shift register when clock signal P1 rises. Then, during



(b) Hand sketch of the layout of one shift register cell.







ANNEXE G

MACROS

INTRODUCTION

Cette annexe contient :

- le mode d'emploi du prèprocesseur de macros
- le texte PASCAL des procedures composant le prèprocesseur de macros

1. Mode d'emploi

Les spécifications d'utilisation du prèprocesseur de macro se trouvent dans le point 9 du manuel utilisateur, pages 89 à 91.

2. Texte PASCAL des procédures du prèprocesseur

```
PROGRAM PREPROCESSEUR(macro,input,output,resultat,origine,interim,reserve);

label 1;

(* sortie directe  en cas d'erreur *)

const  BUFF1 = 10;
       BUFF2 = 500;

(* constantes contenant le nombre maximum de caracteres des 2 chaines ,
   BUFF1 celui de la chaine a remplacer
   BUFF2 celui de la chaine remplacante
*)

type ARRAYBUFFER1 = array [1..BUFF1] of char;
     ARRAYBUFFER2 = array [1..BUFF2] of char;

(* definition des types pour les buffers de caracteres des 2 chaines ,
   ARRAYBUFFER1 celui de la chaine a remplacer
   ARRAYBUFFER2 celui de la chaine remplacante
*)

var BUFFER1 : ARRAYBUFFER1;
    BUFFER2 : ARRAYBUFFER2;
    NBCAR1,NBCAR2,NUMEROMACRO,NUM : integer;
    KOT1,KOT2,KOT3,KOT4 : boolean;
    MACRO,ORIGINE,INTERIM,RESULTAT,RESERVE : text;
    CH : char;

(*
    BUFFER1 - buffer de la chaine a remplacer
    BUFFER2 - buffer de la chaine remplacante
    NBCARREC - nombre de caracteres reconnus
    NBCAR1 - nombre de caracteres deja rencontres pour la chaine a remplacer
    NBCAR2 - nombre de caracteres deja rencontres pour la chaine remplacante
    NUMEROMACRO - numero de la macro traitee
    KOT1 - indicateur de la rencontre du premier "kot" d'une definition
    KOT2 - indicateur de la rencontre du deuxieme "kot" d'une definition
    KOT3 - indicateur de la rencontre du troisieme "kot" d'une definition
    KOT4 - indicateur de la rencontre du quatrieme "kot" d'une definition
    INTERIM - fichier intermediaire
    MACRO - fichier contenant les definitions de macro
    RESULTAT - fichier contenant le texte au kilometre ou les chaines ont
        ete remplacee
    ORIGINE - fichier contenant le texte au kilometre source
    RESERVE - fichier de transition contenant le contenu du fichier source
    CH - caractere
*)
(*****************************************************************************)

procedure ERREUR(NUM : integer);

(* gestion des erreurs *)

begin
  case NUM of
       1 : begin
             writeln('LE NOMBRE DE CARACTERES DE LA CHAINE A REMPLACER ');
             writeln('   DE LA MACRO EST DE PLUS DE 10 CARACTERES');
             write('IL S'''AGIT DE LA MACRO NUMERO : ');
             writeln(NUMEROMACRO);
           end;
       2 : begin
             writeln('LE NOMBRE DE CARACTERES DE LA CHAINE REMPLACANTE ');
             writeln('   DE LA MACRO EST DE PLUS DE 500 CARACTERES');
             write('IL S'''AGIT DE LA MACRO NUMERO : ');
             writeln(NUMEROMACRO);
           end;
       3 : begin
             write('INTRODUCTION INCORRECTE DES "kots" POUR LA DEFINITION');
             write(' DE LA MACRO NUMERO : ');
             writeln(NUMEROMACRO);
           end;
    end;
  goto 1;
end;
(*****************************************************************************)

procedure  REMPLIRBUFF1(var NBCAR1 : integer);

(* procedure qui a comme but d'ajouter un caractere au buffer 1 *)

begin
  NBCAR1 := NBCAR1 + 1;
  if NBCAR1 > 10 then ERREUR(1)
                 else BUFFER1[NBCAR1] := CH;
end;
(*****************************************************************************)
```

```
procedure  REMPLIRBUFF2( var NBCAR2 : integer);

(* procedure qui a comme but d'ajouter un caractere au buffer 2 *)

begin
  NBCAR2 := NBCAR2 + 1;
  if NBCAR2 > 5ØØ then ERREUR(2)
                  else BUFFER2[NBCAR2] := CH;
end;
(*****************************************************************************)

procedure  SUBSTITUER(NBCAR1,NBCAR2  : integer);

(*
   procedure substituant toutes les occurance de la chaine a remplacer, chaine
     se trouvant dans BUFFER1, le nombre de caracteres a reconnaitre est NBCAR1 :

   par une chaine se trouvant dans BUFFFER2, le nombre de caracteres a recon-
   naitre est NBCAR2;
*)

var TROUV : boolean;
    I,NBCARREC,NBCARECRIT,LONG : integer;
    CAR : char;

(* TROUV : indicateur de decouverte d'un bout de chaine a remplacer
   I : compteur
   NBCARREC : nombre de caracteres reconnus dans une chaine a remplacer
   NBCARECRIT : nombre de caracteres copies dans le fichier de la chaine
        remplacante
   LONG : nombre de caracteres a inscrire avant un <CR>
   CAR : caractere lu dans le fichier origine
*)

procedure SUBS1(NBCAR1,NBCAR2 : integer);

var BOOL : boolean;

begin
  TROUV := true;
  NBCARREC := Ø;
  BOOL := false;
  reset(ORIGINE);
  rewrite(INTERIM);
  read(ORIGINE,CAR);
  while not EOF(ORIGINE) do
    begin
      while not EOLN(ORIGINE) do
        begin
          if Ø = NBCARREC then TROUV := true;
          if TROUV = true then nbcarrec := nbcarrec + 1;
          if (BUFFER1[NBCARREC] = CAR) then
            begin
              TROUV := true;
              if NBCARREC = NBCAR1 then
                begin
                  NBCARECRIT := 1;
                  LONG := 7Ø;
                  writeln(INTERIM);
                  while NBCARECRIT <= NBCAR2 do
                    begin
                      write(INTERIM,BUFFER2[NBCARECRIT]);
                      nbcarrec := Ø;
                      LONG := LONG - 1;
                      NBCARECRIT := NBCARECRIT + 1;
                      if LONG <= Ø  then
                        begin
                          LONG := 7Ø;
                          WRITELN(INTERIM);
                          end;
                    end;
                end;
            end
            else
            begin
              TROUV := false;
              if NBCARREC > 1 then
                begin
                  I := 1;
                  while NBCARREC > 1 do
                    begin
                      write(INTERIM,BUFFER1[I]);
                      I := I + 1;
                      NBCARREC := NBCARREC - 1;
                    end
                end ;
              write(INTERIM,CAR);
              if BOOL then
                begin
                  BOOL := false;
                  writeln(INTERIM);
                end;
            end;
          read(ORIGINE,CAR);
        end;
      readln(ORIGINE);
      BOOL := true;
    end;
end;
```

```
procedure SUBS2;

var BOOL : boolean;

(* procedure copiant le contenu du fichier INTERIM dans le fichier ORIGINE *)

begin
  BOOL := false;
  reset(INTERIM);
  rewrite(ORIGINE);
  read(INTERIM,CAR);
  while not EOF(INTERIM) do
    begin
      while not EOLN(INTERIM)  do
        begin
          write(ORIGINE,CAR);
          if BOOL then begin writeln(ORIGINE); BOOL := false end;
          read(INTERIM,CAR);
        end;
      BOOL := true;
      readln(INTERIM);
    end;
  reset(ORIGINE);
end;

begin
  SUBS1(NBCAR1,NBCAR2);
  SUBS2;
end;
(*****************************************************************************)

procedure TRANS1;

(* procedure copiant le contenu du fichier ORIGINE dans le fichier RESERVE *)

var BOOL : boolean;
    CAR : char;

begin
  BOOL := false;
  reset(ORIGINE);
  rewrite(RESERVE);
  read(ORIGINE,CAR);
  while not EOF(ORIGINE) do
    begin
      while not EOLN(ORIGINE)  do
        begin
          write(RESERVE,CAR);
          if BOOL then begin writeln(RESERVE); BOOL := false end;
          read(ORIGINE,CAR);
        end;
      BOOL := true;
      readln(ORIGINE);
    end;
end;
(*****************************************************************************)

procedure TRANS2;

(* procedure copiant le contenu du fichier ORIGINE dans le fichier RESULTAT *)

var BOOL : boolean;
    CAR : char;

begin
  BOOL := false;
  reset(ORIGINE);
  rewrite(RESULTAT);
  read(ORIGINE,CAR);
  while not EOF(ORIGINE) do
    begin
      while not EOLN(ORIGINE)  do
        begin
          write(RESULTAT,CAR);
          if BOOL then begin writeln(RESULTAT); BOOL := false end;
          read(ORIGINE,CAR);
        end;
      BOOL := true;
      readln(ORIGINE);
    end;
end;
(*****************************************************************************)
```

```
procedure TRANS3;

(* procedure copiant le contenu du fichier RESERVE dans le fichier ORIGINE *)

var BOOL : boolean;
    CAR : char;

begin
  BOOL := false;
  reset(RESERVE);
  rewrite(ORIGINE);
  read(RESERVE,CAR);
  while not EOF(RESERVE) do
    begin
      while not EOLN(RESERVE)  do
        begin
          write(ORIGINE,CAR);
          if BOOL then begin writeln(ORIGINE); BOOL := false end;
          read(RESERVE,CAR);
        end;
      BOOL := true;
      readln(RESERVE);
    end;
end;

(*****************************************************************************)

begin
  TRANS1;
  reset(ORIGINE);
  reset(MACRO);
  NBCAR1 := Ø;  NBCAR2 := Ø;
  NUMEROMACRO := 1;
  KOT1 := false; KOT2 := false; KOT3 := false; KOT4 := false;
  read(MACRO,ch);
  while CH <> '"' do
    begin
      while not EOF(MACRO) do
        begin
          while not EOLN(MACRO) do read(MACRO,CH);
          readln(MACRO);
        end;
    end;
  while not EOF(MACRO) do
    begin
      while not EOLN(MACRO) do
        begin
          if ch = '"' then
            begin
              if ((not KOT1) and (not KOT2) and (KOT3) and (not KOT4)) then
                begin KOT3 := false; KOT4 := true; end;
              if ((not KOT1) and (KOT2) and (not KOT3) and (not KOT4)) then
                begin KOT2 := false; KOT3 := true;end;
              if ((KOT1) and (not KOT2) and (not KOT3) and (not KOT4)) then
                begin KOT1 := false; KOT2 := true;end;
              if ((not KOT1) and (not KOT2) and (not KOT3)  and (not KOT4))
                then KOT1 := true;
              if KOT4 then
                begin
                  SUBSTITUER (NBCAR1,NBCAR2);
                  KOT4 := false;
                  NUMEROMACRO := NUMEROMACRO + 1;
                  NBCAR1 := Ø;NBCAR2 := Ø;
                end;
            end
             else
            begin
              if KOT1 then REMPLIRBUFF1(NBCAR1);
              if KOT3 then REMPLIRBUFF2(NBCAR2);
            end;
          read(MACRO,CH);
        end;
      readln(MACRO);
    end;
  if ((CH = '"') and (KOT3)) then
    begin
      KOT3 := false;
      SUBSTITUER (NBCAR1,NBCAR2);
      NUMEROMACRO := NUMEROMACRO + 1;
    end;
  if (KOT1 or KOT2 or KOT3 or KOT4) then ERREUR(3);
  TRANS2;
  TRANS3;
1 : ;
end.
```

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX (NAMUR)

INSTITUT D'INFORMATIQUE

REALISATION D'UN FORMATEUR

mémoire présenté par

Claude Mathieu

et

Alain Schmitz

en vue de l'obtention

du titre de

Licencié et maître en informatique

Année académique 1982-1983

Avant propos

Nous remercions les personnes qui à titre divers, contribuèrent à la réalisation de ce mémoire.

Nous tenons à remercier monsieur Ph. Van Bastelaer qui a dirigé ce mémoire et qui a permis par ses conseils de mener à bien ce travail.

Nous tenons également à témoigner notre gratitude à Monsieur J.D. Nicoud de nous avoir accueillis pour un stage à l'Ecole Polytechnique Fédérale de Lausanne.

Nous remercions vivement l'ensemble des membres du Laboratoire de Micro-informatique pour leur accueil au sein de leur département, et en particulier Monsieur R. Hersch, dont les conseils nous ont permis de mener à bien la réalisation du projet.

Nous tenons enfin à remercier Mademoiselle B.Scoyer pour sa constante disponibilité.

INTRODUCTION

INTRODUCTION

Nous présentons dans ce mémoire la réalisation d'un outil de traitement de texte : un formateur de texte.

La révolution informatique de ces dernières années (tant au niveau matériel qu'au niveau logiciel) a ouvert de nouveaux horizons dans le traitement de texte notamment dans le domaine de la bureautique. Les systèmes Editeur - Formateur remplacent peu à peu les machines à écrire. Ils permettent non seulement d'imprimer du texte mais également du graphisme et des formules mathématiques. Le système que nous avons conçu, baptisé "FTTL" (formatage traitement texte laser) traite du texte avec possibilité d'insertion de graphisme.

Au niveau conceptuel, ce mémoire comprend deux parties essentielles :

- la présentation générale d'une étude de systèmes existant dans le domaine du formatage de texte

- la conception et la réalisation d'un formateur particulier

Au niveau structurel, ce mémoire se compose de 7 chapitres :

- dans un premier chapitre, nous faisons un historique du problème du formatage en présentant les différents formateurs représentatifs de l'évolution

- dans un deuxième chapitre, nous décrivons la configuration du matériel existant sur le lieu de notre stage à l'Ecole Fédérale Polytechnique de Lausanne (E.P.F.L.). C'est en fonction de l'étude réalisée au chapitre 1, du matériel disponible et des besoins de l'E.P.F.L. que nous nous sommes fixé les objectifs et les caractéristiques du formateur à réaliser

- dans les troisième et quatrième chapitres, nous décrivons la méthode d'analyse qui a amené la réalisation du formateur, plus spécifiquement, nous présentons l'analyse fonctionnelle et l'analyse organique. L'analyse fonctionnelle décrit le "QUE" du problème (on spécifie ce que l'on va faire). L'analyse organique traite le "COMMENT" (on présente une architecture physique du système, les spécifications des données et des traitements, le choix d'un langage de programmation et les méthodes de tests utilisées)

- dans le cinquième chapitre, nous présentons le manuel utilisateur du formateur.

- dans le sixième chapitre, nous faisons une étude comparative entre le formateur réalisé et deux autres systèmes que nous avons dû couramment employer :

  - le formateur "JUSTIF" qui nous a servi de modèle (formateur utilisé à l'Ecole Polytechnique de Lausanne)

  - le formateur "NROFF" du système "UNIX" que nous avons employé pour rédiger le mémoire

- dans le dernier chapitre du mémoire, nous présentons les problèmes (de conception, de matériel) rencontrés sur le lieu du stage et aux Facultés, l'expérience apportée par le stage ainsi que les modifications à apporter afin que le système soit adaptable aux Facultés. Enfin nous évaluons le travail effectué.

Nous utilisons trois conventions de rédaction :

- numérotation décimale pour les chapitres

- la première occurrence de chaque mot technique sera suivie de (*) le référant à une liste de définitions classée par ordre alphabétique et se trouvant à la fin du rapport

- chaque fois que nous faisons une référence bibliographique dans le texte, nous insérons le nom de l'auteur suivi de la date d'édition ; la liste des références complètes se trouve à la fin du rapport, classée par ordre alphabétique sur le nom de l'auteur

A ce rapport seront joints une annexe et un manuel utilisateur :

ANNEXE : elle contient un complément à l'analyse organique qui reprend les spécifications de tous les modules de programmation, les programmes PASCAL, une description détaillée du format "FDD", une description de l'imprimante laser CANON LBP10, les programmes de décodage/encodage avec leurs spécifications, des exemples de feuilles imprimées par cette imprimante et le programme de traitement de macros avec ses spécifications.

MANUEL UTILISATEUR : il contient une introduction à la description du formateur, la description des commandes du niveau global et du niveau local, la classification des commandes par ordre alphabétique, la liste des erreurs détectées, les types de fontes et de polices disponibles, deux exemples d'utilisation et un mode d'emploi d'utilisation des macros. Le manuel utilisateur a été rédigé à l'aide du formateur que nous avons conçu.

TABLE DES MATIERES

Table des matières

Chapitre 1 :

SYSTEMES DE FORMATAGE. DESCRIPTION ET EVOLUTION

Chapitre 1:

SYSTEMES DE FORMATAGE. DESCRIPTION ET EVOLUTION

1.1. Introduction.

Deux tâches constituent principalement la préparation d'un document. Tout d'abord la définition du contenu et de la structure du document et ensuite la génération du document à partir de ses spécifications [Nicoud 82].

La première tâche est le problème de l'édition tandis que la seconde est le problème du formatage. Plus précisément, le formatage concerne la disposition des éléments (paragraphe, entête, liste,...) d'un document afin que ce dernier puisse être visualisé sur imprimantes, écrans vidéos, ...

Le problème du formatage n'a été que très récemment étudié. Les raisons du développement de ces systèmes sont à la fois économiques et technologiques : l'augmentation du coût de la main-d'oeuvre (pour la mise en forme d'un document manuellement), la diminution du coût des ordinateurs et l'amélioration technique du matériel disponible (imprimantes graphiques).

Ce chapitre aura comme but de définir le problème du formatage et d'évaluer la plupart des systèmes de formatage représentatifs de l'évolution.

1.2. Problèmes du formatage

Les principales fonctions d'un formateur sont les suivantes :

- la sélection des éléments de base concret (par exemple le paragraphe, le chapitre, l'équation, la table, la figure, l'entête de page, la note bas de page) et pour chacun d'eux, la détermination des caractéristiques typographiques (*).

- le placement horizontal et vertical des éléments (par exemple, les opérations de décalage par rapport aux marges)

- l'alignement horizontal et vertical des éléments les uns par rapport aux autres (par exemple le décalage d'un paragraphe par rapport à l'entête du chapitre)

- le "flottement", où la détermination d'un écart optimal entre éléments pour l'impression

1.3. Systèmes représentatifs de l'évolution

1.3.1. Introduction

L'évolution et l'historique du problème du formatage seront étudiés dans cette partie. Quelques systèmes représentatifs de l'évolution illustreront cette étude.

Une distinction est à faire entre ce que nous allons appeler les "vrais" formateurs et les éditeurs-formateurs. Un "vrai" formateur (ou formateur "batch" (*)) accepte en entrée un texte source (souvent un texte au kilomètre (*)) précédemment préparé par un système d'édition séparé.

Les éditeurs-formateurs intégrés permettent de visualiser un document lors de la création et de la modification sans devoir quitter le système. En d'autres termes, l'édition, le formatage et la visualisation sont combinés dans un seul système.

Le sujet de ce mémoire étant la réalisation d'un formateur, nous allons nous limiter à donner un aperçu sur les "vrais" formateurs. A chacun de ceux-ci, un exemple d'utilisation sera présenté. Il aura comme but de réaliser le traitement du texte de la figure 1.1. Les exemples sont donnés seulement à titre indicatif.

```
                    Exemple

Un système de traitement de texte est composé de
différentes étapes.

Ces étapes sont :

        . l'édition
        . le formatage
        . la visualisation

Le but de ce mémoire est de traiter le problème
du formatage.
```

Fig. 1.1. Exemple de texte formaté.

1.3.2. Première génération de formateur.

1.3.2.1. Introduction

Les premiers formateurs apparurent vers le milieu des années 1960. Les fonctions de formatage fournissaient, à cette époque, très peu de fonctions. La plupart des fonctions étaient associées au concept de ligne et de page du document. Très peu de fonctions étaient associées au contenu logique du document : mot, phrase, paragraphe. Le langage de formatage était fixe et non extensible. Il n'était pas possible de structurer le document.

Pour ces premiers systèmes, les commandes de formatage étaient insérées dans le texte à formater, cette forme est d'ailleurs toujours utilisée dans les formateurs récents.

Afin de distinguer les commandes du texte, deux possiblités ont été développées. La première est de distinguer les lignes de commandes des lignes de données par un caractère spécial se situant dans la première colonne de la ligne. Ceci a été utilisé par le système RUNOFF. ne ligne de commande commence par "." ou par "'"). La seconde, utilisée par le système FORMAT, est de faire précéder une commande par un caractère reservé. La fin de la commande est indiquée soit par un séparateur, soit - si les commandes ont une longueur uniforme - après un certain nombre de caractères.

Les premiers formateurs ont également introduit l'utilisation de caractères réservés afin de signaler une action "limitée", par exemple dans le système FORMAT, le " " forcera le caractère suivant en majuscule.

L'usage de macros n'était pas assuré et il n'était pas possible de modifier les actions d'une commande particulière.

1.3.2.2. RUNOFF

RUNOFF, apparut en 1964 sur CTSS (Compatible Time Sharing System) au MIT [Saltzer 65]. Son éditeur séparé s'appelle TYPSET.

RUNOFF travaille sur une description d'un document encodé en caractères majuscules et/ou minuscules. Il produit un texte formaté pour une imprimante orientée ligne (type machine à écrire).

La première version de RUNOFF contenait 18 primitives, toutes orientées pour formater les pages du document.

Une liste de fonctions améliorant le formatage du texte pour un utilisateur de RUNOFF (liste non exhaustive) :

- pour une ligne : les fonctions de centrage, d'indentation, de cassure
- pour un groupe de lignes : les fonctions de détermination de la longueur de la ligne, justification, indentation
- pour l'arrangement des lignes : les fonctions de détermination de la largeur du blanc, du double blanc, de la tabulation
- pour le fichier : la fonction de liaison avec un autre

Toutes ces fonctions traitent le format physique du document, aucune ne traite le format logique.

Une ligne est soit dans une ligne de commande (précédée de "." dans la première colonne), soit dans un ligne de texte.

La figure 1.2. illustre l'utilisation de RUNOFF. Cet exemple a pour but de réaliser le formatage de la figure 1.1. qui se trouve dans l'introduction (au point 1.3.1.).

```
.center
Exemple
.space 2
Un système de traitement de texte est
compose de différentes étapes.
.nojust
.space 1
Ces étapes sont
.indent 10
.undent 2
- l'édition
.space 1
.undent 2
- le formatage
.space 1
.undent 2
- la visualisation
.indent 0
.space 1
.adjust
Le but de ce mémoire est de traiter le
problème du formatage.
```

Remarque :

Une ligne de commande commence par un point.

Fig. 1.2. Exemple d'utilisation de RUNOFF

1.3.2.3 FORMAT

FORMAT fut développé et utilisé sur ordinateur IBM S/360 [Berns 68] [Berns 69] [Berns et Ehrman 71]. La première description apparut vers la fin des années 1960. Un éditeur était fourni et bien qu'il se trouvait dans le même programme "physique" que le formateur, les fonctions d'édition étaient distinctes.

FORMAT tourne sous une forme batch. Le texte source se trouve sur cartes perforées. Le texte source est entièrement en caractères majuscules et les caractères qui ne sont pas explicitement "dis" en majuscules sont convertis en minuscules. Le document concret contient donc des caractères majuscules et minuscules.

Comme dans le cas de RUNOFF, la plupart des commandes manipulent des objets physiques orientés page :

- groupe de lignes : fonctions de remplissage, de définition de la longueur de la ligne, de justification
- page : fonctions de cassure, numérotation, détermination de la longueur, plusieurs colonnes, spécification des notes et des entêtes

FORMAT commence à considérer certains éléments logiques :

- mot : fonction produisant une liste alphabétique des mots
- phrase : fonction de soulignement, fonction de centrage, ...
- paragraphe : fonction d'indentation,...

En différence avec le traitement des caractères de RUNOFF, certaines opérations de FORMAT peuvent s'appliquer à un seul caractère.

FORMAT utilise le système d'insertion des commandes dans le texte source sans devoir passer à la ligne.

La figure 1.3. illustre l'utilisation de FORMAT. Cet exemple a pour but de réaliser le formatage de la figure 1.1. qui se trouve dans l'introduction (au point 1.3.1.).

```
)V
INDENTATION OF COLUMNS ON LEFT AND RIGHT
IS (5,5), (10,5)
PARAGRAPH INDENT IS 0 PRINT POSITIONS
SEPARATATION LINES BETWEEN PARAGRAPHS ARE 1
TABS ARE SE AT RELATIVE COLOMN POSITION 5
NONTRIVIAL BLANK IS REPRESENTED B4 SPECIAL
CHARACTER 44 (#)
(2)
)M¢ ¢EXEMPLE )M¢LLP CUN SYSTEME DE TRAITEMENT
 DE TEXTE EST COMPOSE DE DIFFERENTES ETAPES
)P ¢CES ETAPES SONT :
)LLH2W1 ##-)T L EDITION
)LLH2W1 ##-)T LE FORMATAGE
)LLH2W1 ##-)T LA VISUALISATION
)HLLH2W2 ##-)T ¢LE BUT DE CE MEMOIRE EST DE
TRAITER LE PROBLEME DU FORMATAGE
```

Remarques :

Les lignes suivant le symbole ")V" jusqu'à la ligne contenant le mot GO, sont des lignes définissant les attributs à associer aux paragraphes.
Chaque symbole ")" dans le texte détermine une instruction à effectuer, c'est ainsi que :

- ")P" spécifie un début de paragraphe
- ")L" spécifie une fin de paragraphe
- ")M" spécifie que ce qui suit doit être centré
- "(¢" spécifie que le caractère suivant sera en majuscule
- ")¢" spécifie que les caractères de la phrase suivante seront en majuscules
- "#" spécifie que le caractère blanc est significatif

Fig.1.3. Exemple d'utilisation de FORMAT

1.3.3. Premiers formateurs structurés.

1.3.3.1. Introduction

De la fin des années 60 au milieu des années 70, une nouvelle génération de formateur basée sur les leçons tirées de l'utilisation de la première génération apparut.

La description du document resta à peu près la même. Cependant, les fonctions s'améliorèrent aussi bien quant à leur nombre que quant à leur sophistication.

Des macros assurèrent la possibilité de regrouper un ensemble de commandes souvent utilisées, afin de définir de nouvelles commandes et afin de refléter la structure logique du document à formater.

Pour faciliter le travail de l'utilisateur, de nouvelles fonctions furent crées :

- numérotage automatique des chapitres
- tables d'indices créées durant le formatage
- placement et numérotation de notes bas de page
- ...

A ce stade apparut l'idée qu'un formateur devait être plus qu'une simple mise en place de lignes et de mots sur une page. Un document est constitué d'une suite d'éléments logiques (mot, phrase, paragraphe) et le but d'un formateur doit être de manipuler ces objets.

Une autre différence avec les premiers formateurs résulte de la sophistication des imprimantes (possiblité de nouvelles fonctions tel le changement de police,..) et les appareils permettant l'encodage (écran vidéo,...). Assurer le moyen de créer un texte en entrée ne fut plus considéré comme un problème relevant du formatage, mais un éditeur de texte indépendant assure cet encodage. Cependant, le "package" de formatage comprend toujours les programmes d'édition et de formatage.

1.3.3.2. PUB

PUB fut développé au laboratoire d'Intelligence Artificielle de Standford, au début de 1971, pour être utilisé sur PDP-10 [Tesler 72]. Ses constructeurs l'appelèrent "The Document Compiler" , ce qui illustre le parallélisme entre la traduction d'un texte source abstrait en un texte imprimable et la compilation d'un programmme, traduction d'un programme dans un langage en un programme exécutable.

Deux types de commandes sont utilisés par PUB. Les commandes de bas niveau qui sont du même type que celles utilisées par FORMAT (orientées ligne, page, mot, phrase, paragraphe). En plus de ces commandes de bas niveaux, PUB fournit des commandes de haut niveau dont :

- plusieurs colonnes sur une page
- chapitres, sous-chapitres, ... numérotés automatiquement
- entêtes de chapitre pouvant être utilisés pour former une table des matières
- plusieurs caractères de base pouvant être imprimés au même endroit afin de former de nouveaux caractères
- référence à un autre fichier

Les constructeurs de PUB ont fait une tentative de classifier certains éléments. Un paragraphe est constitué de 3 parties appelées respectivement : "crown", "vest", "hem". "Crown" correspond à la première ligne du paragraphe (ligne indentée), "vest" correspond au corps du paragraphe (texte dont on a retiré la première ligne), "hem" correspond à la dernière ligne du paragraphe.

Une page de PUB est constituée de "areas" (régions). Une région est soit d'un type qui peut être continué à la page suivante (type = "texte"), soit d'un type qui doit être entièrement sur la page courante (type = "titre"). Une région doit avoir un nom et doit être positionnée arbitrairement sur une page. Une page devant contenir au moins une région de type "texte".

PUB permet en outre l'utilisation de macros (macros pouvant avoir une définition récursive).

La principale contribution de PUB consiste en l'incorporation de blocs de structure (blocs encadrés par les instructions "begin" et "end"). En plus, l'utilisation de macro permet d'étendre le langage de commande ; les variables et les macros définies au sein d'un bloc reprennent leur(s) valeur(s) intiale(s) dès la rencontre du "end".

Les commandes de PUB ont la même forme que celle de RUNOFF.

Quelques symboles et séquences de symboles ont des significations spéciales dans les lignes de texte.

La figure 1.4 illustre l'utilisation de PUB. Cet exemple a pour but de réaliser le formatage de la figure 1.1. qui se trouve dans l'introduction (au point 1.3.1.).

```
.TURN ON "↓","_","#"
.SINGLE SPACE
.INDENT 0
.PREFACE 1
.ONCE CENTER
Exemple

Un systeme de traitement de texte est compose de
differentes etapes

.SKIP
.BEGIN INDENT 3,5,5 ; PREFACE 1;


-#l'edition
-#le formatage
-#la visualisation

.END
.SKIP 1

Le but de ce memoire est de traiter le probleme
du formatage
```

Remarques :

Une ligne de commande commence par un point. Plusieurs commandes peuvent être placées sur une même ligne, séparée par une virgule, ceci afin d'éviter les ambiguités. Une commande peut aussi être insérées dans le texte, dans ce cas elle sera entre les caractères "{", "}".
Chaque paragraphe commence par une ligne blanche.
La commande "ONCE CENTER" a comme but de faire centrer le paragraphe qui suit.
Le caractère "#" représente un caractère blanc significatif, le "↓_" le début d'un soulignement et "_↓" la fin de celui-ci.

Fig. 1.4. Exemple d'utilisation de PUB

1.3.3.3. NROFF

NROFF est le système de formatage utilisé sur le système UNIX.[Ossanna 74] Il fut développé au milieu des années 70 sur PDP-11 et dérive de ROFF [Thompson et Ritchie 75] qui, lui-même, était dérivé de RUNOFF.

Vu l'utilisation des macros, NROFF permet beaucoup de choses, mais nous nous limiterons ici à parler des instructions de base (de nombreux pré et post processeurs ont été développés en effet pour l'usage de NROFF).

Les éléments supportés par NROFF sont pratiquement les mêmes que ceux supportés par PUB : ligne, page, mot, phrase, paragraphe, surimpression de caractères pour en former de nouveaux.

NROFF fournit des "environnements" qui, similaires au bloc défini dans PUB, permettent le regroupement de certains paramètres. Il est posible de créer à un moment donné un nouvel "environnement" et ultérieurement de restaurer l'ancien.

Des registres numériques pré-définis fournis par le système peuvent être inclus dans le texte (exemple : pour numéroter les chapitres, les pages, ...).

Des macros peuvent être définies et peuvent être appelées récursivement (jusqu'à neuf paramètres peuvent être invoqués simultanément). Des commandes conditionnelles peuvent être également disponibles.

Il est intéressant de distinguer ce qui est fourni explicitement dans les commandes de base et ce qui ne l'est pas. Ce qui N'EST PAS fourni : notes bas de page, annotations en entête de page, colonne multiple, annotations de bas de page. Des mécanismes combinés avec l'usage des macros peuvent être utilisés cependant pour implémenter facilement ces fonctions.

Le langage de formatage consiste à séparer les commandes et le texte. Deux groupes de commandes sont fournis.

- dans le premier, une commande se trouve sur une ligne séparée du texte (système de RUNOFF), dans ce cas, on distingue une ligne de commande d'une ligne de texte en regardant si le premier caractère de la ligne est un ".", un "'" ou autre chose

- dans le deuxième, une commande commence par un boa (*). Ce groupe fournit les mêmes sortes de fonctions que les caractères spéciaux dans PUB et dans FORMAT.

Le langage de base de NROFF est de très bas niveau et très difficile à utiliser. Beaucoup de ces commandes sont utilisées par un formateur pré-processeur.

L'utilité de NROFF réside dans la grande possibilité qu'il a à

s'adapter aux exigences de l'utilisateur grâce à l'usage de pré et post processeurs.

La figure 1.5 illustre l'utilisation de NROFF. Cet exemple a pour but de réaliser le formatage de la figure 1.1. qui se trouve dans l'introduction (au point 1.3.1.).

```
.ce 1
Exemple
.sp 1
Un systeme de traitement de texte est compose de
differentes etapes
.sp 1
Ces etapes sont :
.sp 1
.in +5
.ti -2
-\l'edition
.sp 1
.ti -2
-\le formatage
.sp 1
.ti -2
-\la visualisation
.in
.sp 1
Le but de ce memoire est de traiter le probleme du
formatage
```

Remarques :

Une ligne de commande commence par un point.
Le caractère "\" est utilisé pour donner une signification spéciale au caractère qu'il suit.
L'instruction ".in +5" a comme but d'incrémenter l'indentation de 5 caractères et ".in" de restaurer la valeur précédente

Fig. 1.5. Exemple d'utilisation de NROFF.

1.3.4. Formateurs structurés avec de nombreuses fonctions.

1.3.4.1. Introduction

SCRIBE, TEX, et le formateur fourni par le système UNIX sont des formateurs fortement structurés qui offrent de nombreuses fonctions.

Le nombre de fonctions de ces systèmes a été considérablement augmenté par rapport aux formateurs précédents. TEX et le système de formatage de UNIX permettent l'inclusion d'équations mathématiques dans le document.

TEX est vu comme un système où l'utilisateur désire positionner exactement ces éléments sur la page à imprimer. SCRIBE est vu comme un système où l'utilisateur doit facilement spécifier les éléments abstraits au sein du document. Pour ce formateur, le positionnement des éléments sur la page est laissé au système. TEX à des commandes peu flexibles, peu extensibles, tout au contraire de SCRIBE.

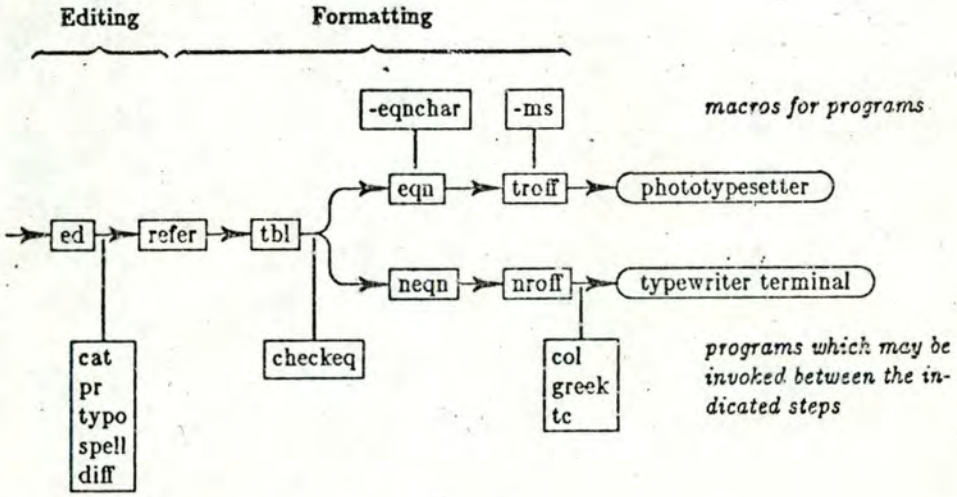
Chacun de ses systèmes fournit des fonctions d'optimisation.

1.3.4.2. UNIX : outil de formatage

Le système de formatage de UNIX comprend un grand nombre d'outils [Kernighan, Lesk et Ossanna 78]. Il a été développé à partir de NROFF et de TROFF. Un grand nombre de packages de macros, de pré et post processeurs ont été développés. Il permet, en outre, de formater du texte, des tables, des équations mathématiques, du graphisme.

Le but est de fournir un grand nombre d'utilitaires tout en ayant le minimum de commandes de base.

La figure 1.6. illustre les différents packages utilisés par le système UNIX tandis que la figure 1.5 illustre l'utilisation du système de formatage de UNIX (qui revient à utiliser NROFF quand seul un texte sans formule, graphique, tableau, ... doit être formaté).
La figure provient du livre de J. Nievergelt, G. Coray, J.-D. Nicoud, A.C. Shaw "Document Preparation Systems, page 155 [Nicoud 82].



- *ed*: line-oriented text editor
  - *cat*: list file without pagination
  - *pr*: list file and paginate for printing
  - *typo*: detect spelling errors using statistical analysis
  - *spell*: detect and attempt to correct spelling errors with dictionary
  - *diff*: compare files, generate troff commands to place marginal bars when differences found
- *refer*: generate bibliographic citations. Refer has its own separate subsystem for maintaining the bibliography data base file.
- *tbl*: table formatter
- *eqn* and *neqn*: mathematical equation formatters
  - *checkeq*: make sure that equation is syntactically correct before passing it on to eqn or neqn
  - *eqnchar*: macro package specifying special characters normally unknown in troff
- *troff* and *nroff*: RUNOFF-like formatters
  - *ms*: macros for partial separation of content from format
  - *col*: convert nroff output to print on devices without reverse scrolling
  - *greek*: convert nroff output to print Greek characters on Teletype 37
  - *tc*: convert troff output to print on Tektronix 4024 DVST terminal

Fig. 1.6. UNIX : système de formatage

1.3.4.3. SCRIBE

SCRIBE fut développé vers la fin des années 70 par B. Reid à l'Université de Carnegie-Mellon [Reid 80a] [Reid 80b] [Reid et Walker 80] [Reid 81]

L'approche est assez différente de celle des précécents formateurs. L'impression finale et la disposition ne sont en effet plus directement spécifiées par l'utilisateur, mais bien par le programme de formatage.

Le texte source comprend des indications logiques, mais pas d'indications physiques. Les détails de formatage sont spécifiés par le système. L'utilisateur aura donc le devoir de spécifier la structure logique. Un des résultats de cette philosophie est que les éléments seront tous de type "TEXTE".

SCRIBE ne comprend pas de facilités. Exemple : la construction de tables. Aucun élément logique autre que ceux existant ne peut être défini.

Un des avantages de la description de SCRIBE est sa très grande portabilité.

Description des commandes et de l'environnement.

Un document est déclaré dans un type particulier (lettre, article, revue,...) . En fonction de ce type, les attributs de formatage sont spécifiés.

La définition des attributs se trouvera dans une base de données de SCRIBE. Il est possible de retirer ou d'ajouter des types particuliers dans cette base et de leur définir des attributs.

Un mot clé est précédé d'un symbole spécial et suivi d'arguments. Exemple : @LABEL (LABEL-NAME)

Il est possible de créer ou de modifier des définitions pour les environnements, soit à un niveau local, soit à un niveau global.

Possibilités données à l'utilisateur.

De nombreux outils (que l'on trouve également dans PUB et dans UNIX) sont utilisables dans SCRIBE. Ils permettent la création de table des matières, d'index, de la numérotation automatique des chapitres, des notes bas de page, ainsi que le placement de notes bas de page et la génération d'une bibliographie.

SCRIBE permet aussi de découper un long document afin de formater ses diverses parties indépendamment.

La figure 1.7. illustre l'utilisation de SCRIBE. Cet exemple a pour but de réaliser le formatage de la figure 1.1. qui se trouve dans l'introduction (au point 1.3.1.).

```
@Style (indent = 0,spacing = 1,spread + 1)
@Headint (Exemple)

Un systeme de traitement de texte est compose de
differentes etapes

Ces etapes sont :
@Begin (itemize)
l'edition

le formatage

la visualisation

@End(itemize)

Le but de ce memoire est de traiter le probleme
du formatage
```

Remarques :

Une commande est precedee de "@".
La commande @Style modifie les attributs definis pour l'ensemble du document.
Chaque paragraphe est separe par une ligne blanche.
Un environnement specifique est delimite par les commandes "@begin" et "@end", le type de l'environnement est mis entre parentheses.

Fig. 1.7. Exemple d'utilisation de SCRIBE

1.3.4.4. TEX

TEX fut développé par D. Knuth à l'Université de Standford vers la fin des années 70 [Knuth 79].

Son but est d'assurer le formatage d'un document contenant des expressions mathématiques.

Le système permet la visualisation de ce qui sera imprimé.

De nouveaux algorithmes ont été développés pour découper un paragaphe en lignes, pour regouper les lignes en page. Le langage de formatage permet la description précise des objets concrets : texte, tabulateur, élément mathématique.

Les éléments concrets sont représentés sous forme de "boîte" à deux dimensions reliées les unes aux autres par un concept de "colle". Une boîte est définie par une taille (horizontale et verticale). Une boîte contient des caractères, des mots, des lignes, des paragraphes et des pages. La "colle" correspond à l'espace entre les boîtes. Cet espace a une taille fixée qui peut être compressée ou élargie. Ce concept de colle existe aussi entre les mots et les phrases.

Le langage de formatage se présente sous la forme de séquences de contrôle précédé d'un caractère spécial (souvent le boa). Certains autres caractères ont également une signification spéciale. Tous les caractères spéciaux peuvent être redéfinis.

Afin de spécifier une zone ayant des caractéristiques particulières, TEX utilise la notion de "groupe". Un groupe est délimité par l'accolade ouvrante et l'accolade fermante.

Un mode spécial est fourni afin de faciliter la spécification des équations mathématiques. Nous ne le décrirons cependant pas dans le présent chapitre.

Des facilités quant à l'usage de macros sont également fournies. Ces facilités permettent de définir de nouvelles commandes.

TEX utilise le même procédé que celui utilisé par UNIX, il s'agit de séparer les lignes de commande et les lignes de texte.

TEX tente d'éviter l'isolement d'une ligne sur une page, soit en début de page , soit en fin de page.

La figure 1.8. illustre l'utilisation de TEX. Cet exemple a pour but de réaliser le formatage de la figure 1.1. qui se trouve dans l'introduction (au point 1.3.1.).

```
\input basic % defines the standart macros,
  formatting parameters
\parskip 10pt
\parindent 0pt % no indentation
\def\yskip{\vskip3pt}
\def\textindent#1{\noindent
    \hbox to 19pt{\hskip0t plus1000pt
     minus 1000pt#1 }\!}
\def\hang{\hanindent19pt}
\hsize 4in
\ctrline{\bf Exemple}
\vskip 24pt
Un systeme de traitement de texte est compose
de differentes etapes,

Ces etapes sont :
{\parskip 0pt
\par\yskip\texteindent{$\bullet$}\hang l'edition
\par\yskip\texteindent{$\bullet$}\hang le formatage
\par\yskip\texteindent{$\bullet$}\hang la visualisation}

Le but de ce memoire est de traiter le probleme du formatage.

\vfill % fill out rest of page with space
\end
```

Remarque :

Les douze premières lignes établissent les macros et les paramètres de formatage. Le texte commence à la ligne 13.

Fig. 1.8. Exemple d'utilisation de TEX

1.3.5. Editeurs-formateurs intégrés.

Une solution proposée pour le formatage consiste à intégrer l'éditeur au formateur.
Deux catégories existent parmi ces systèmes :

- la première où les éléments utilisés dans le formatage ont été intégrés avec ceux utilisés à l'édition, mais les fonctions d'édition et de formatage ne sont pas intégrées. Dans cette catégorie : QUIDS qui fut crée vers le milieu des années 70 à l'Université de Londres. (QUIDS = Quick Interactive Documentation System) [Colouris 76]

- la seconde, où la plupart des fonctions et des éléments ont été intégrés. Lors d'un changement à l'édition, on peut visualiser directement le document concret.

Quelques-uns de ces formateurs-éditeurs :

- ceux développés par XEROX :

  - XEROX-ALTO, station/ordinateur personnel développé en 1973 [Thacker 79]

  - BRAVO, développé en 1978 [Lampson 78]

  - STAR, annoncé en 1981 [Seybold 81] [Smith 82]

  - SMALLTAK, qui n'est ni un formateur, ni un éditeur ; mais un langage interactif . Le système est basé sur des classes d'objets [Goldberg et Kay 76] [Ingalls 78] [Shock 79] [BYTE MAGASINE 81]

- WANG WORD PROCESSEUR, système de formatage commercialisé vers la fin des années 70

- KATIB et HATTAT, écrit en 1975 par M. Mac Kay au Departement de Classique à l'Université de Washington [MacKay 77]

- TRIX/RED, développé vers 1975 au Lauwrence Livermore Laboratories [Beatty, Chin, et Moll 79]

- IDEAL, développé vers 1980 par C. Van Wyck [Van Wyk 80]

- GML (Generalized Markup Language), developpé par C.F. Goldfarb d'IBM vers 1974 [IBM 80a] [IBM 80b] [Goldfarb 81a] [Goldfarb 81b]

- JANUS, éditeur-formateur en développement chez IBM (laboratoire de recherche de San-Jose) [Chamberlin 81] [Chamberlin 82]

- ETUDE, éditeur-formateur implémenté au MIT [Good 81] [Hammer 81a] [Hammer 81b] [Ilson 80]

- YALE'S PEN en développement à l'Université de Yale [Allen, Nix et Perlis 81]

Chapitre 2 :

CONFIGURATION DU MATERIEL EXISTANT

Chapitre 2:

CONFIGURATION DU MATERIEL EXISTANT

2.1. Introduction

Dans ce chapitre, nous présentons le matériel sur lequel nous avons implémenté le formateur à L'Ecole Polytechnique Fédérale De Lausanne.

Ce chapitre comprendra :

- la configuration générale de l'architecture
- la description de l'imprimante laser CANON LBP10
- la description de l'imprimante SANDERS
- la description du SMAKY
- la description du réseau COBUS
- la description du terminal VISUAL 200

2.2. Configuration générale de l'architecture

L'architecture informatique de l'Ecole Polytechnique Federale de Lausanne est basée sur un réseau principal (réseau EPNET), sur lequel est connecté un réseau local (réseau COBUS).

En ce qui concerne le réseau EPNET (réseau net-one de la firme UGERMANN-BASS compatible ETHERNET) :

- un ordinateur VAX 32 bits
- 32 terminaux VIDEO V200
- 2 imprimantes : une imprimante rapide LPA120 et une télétype
- 2 unités à bandes et 3 unités à disques
- un serveur d'imprimante

En ce qui concerne le réseau COBUS :

- un ordinateur 32 bits
- 30 terminaux SMAKY
- un serveur d'imprimante
- 3 imprimantes : une SANDERS, une VERSATEC et une CANON LBP-10
- un serveur de fichier
- une plaque mémoire M68000 associé à l'imprimante laser CANON LBP-10
- la connexion EPNET - COBUS

La figure 2.1 (page suivante) illustre ces différents composants.

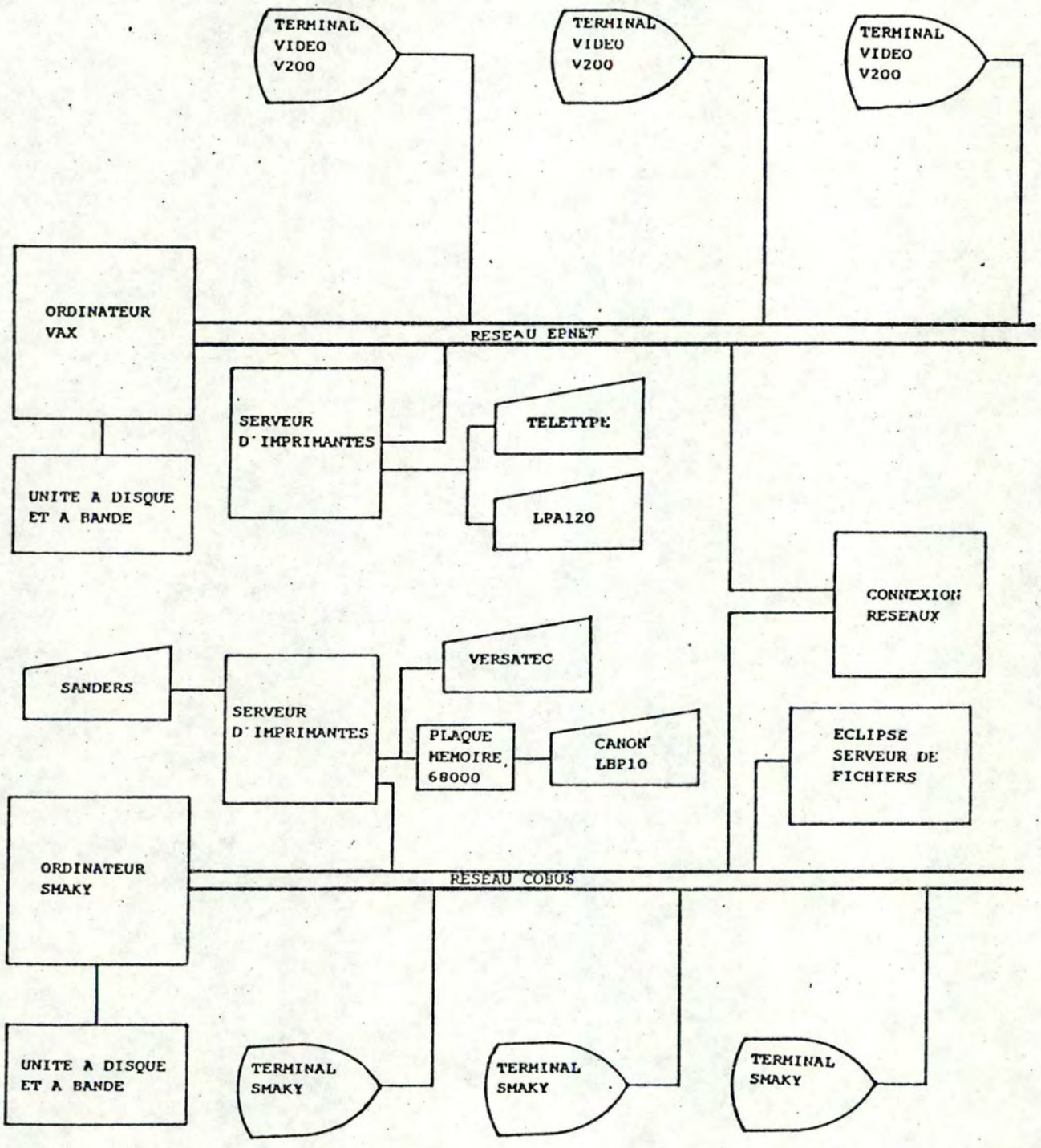


Fig. 2.1. Shèma de l'architecture

2.3. Imprimante Laser LBP-10

Actuellement les services automatisés sont caractérisés par un système distribué où des stations de travail puissantes et intelligentes sont reliés à des serveurs. Parmi ces serveurs, des systèmes d'impression de haute qualité et rapides sont indispensables.

Une imprimante intégrée dans un réseau doit être de haute résolution, dotée d'une grande vitesse d'impression et son prix doit être raisonnable ; les imprimantes performantes, actuellement sont les imprimantes électrostatiques comme la VERSATEC 80 ou les imprimantes à laser.

On peut citer quelques critères de performances :

- résolution
- vitesse d'impression
- vitesse à laquelle une page est générée
- nombre de jeux de caractères disponibles (taille, style)
- possibilités graphiques

L'imprimante laser LBP-10 [NEW ENTERPRISES DIVISION 80] tend à rassembler au mieux ces critères. Elle est décrite quant à sa technologie et quant à ses performances dans l'annexe E.

2.4. L'imprimante SANDERS

La SANDERS [EPFL 81b] est une imprimante travaillant sur une représentation matricielle des caractères. Contrairement à beaucoup d'autres imprimantes qui sont limitées à des matrices fixes, la SANDERS a un degré de contrôle sur le déplacement des points. Avec ce contrôle l'imprimante peut produire des caractères de haute qualité.

Les caractères résultant de l'impression s'obtiennent par plusieurs passes de l'unité d'impression sur la ligne.

La vitesse d'impression dépend essentiellement de deux facteurs :

- la grandeur et le style des caractères
- le nombre de passes associées à la fonte

L'imprimante est contrôlée par un microprocesseur interne qui règle les fonctions mécaniques et manipule les données qui seront imprimées ; il contrôle également les communications entre l'imprimante et l'ordinateur central. Le microprocesseur et les caractères de fonte différents stockés dans une mémoire de type ROM permettent à l'imprimante de changer de fonte à n'importe quel moment sans mouvement de l'unité d'impression.

La vitesse d'impression est :

- pour une passe : 120 à 256 caractères par seconde
- pour quatre passes : 30 à 50 caractères par seconde

2.5. Le SMAKY

Le SMAKY [EPFL 81a] est un système complet comportant :

- un processeur Z80
- 64k bytes de mémoire RAM, 2k bytes de mémoire ROM
- un écran et un clavier

La mémoire ROM contient le système d'exploitation et une librairie de routines. L'écran alphanumérique/graphique de 20 lignes de 64 caractères et 256 * 120 points travaille en mode DMA directement dans la mémoire RAM du système. Le tout est agrémenté d'une multitude d'interfaces avec le monde extérieur (clavier, interfaces série/parallèle, contrôleur de mémoire de masse...).

Trois types de mémoire de masse sont disponibles :

- micro-floppy
- micro-disque
- COBUS

La mémoire de masse COBUS donne la possibilité de dialoguer à plusieurs avec une mémoire de masse importante au travers d'un réseau.

Un système d'exploitation de type UNIX est recherché pour le SMAKY. Des compilateurs pour PASCAL, C, MODULA, PORTAL, BASIC, FORTH, COBOL, FORTRAN sont en phase d'implémentation avec des facilités de développements adéquats, de même que des logiciels graphiques pour l'enseignement et des logiciels de gestion de fichiers.

2.6. Réseau COBUS

Le réseau COBUS [Sommers 80] est un réseau de faible coût qui relie les terminaux et les micro-ordinateurs du laboratoire. Il se compose de vingt stations alphanumériques avec possibilités graphiques et est utilisé pour l'enseignement de la programmation de microprocesseur, éditeur de texte, et des recherches (hard et soft) sur les réseaux locaux. COBUS peut être considéré comme une version "bon marché" du réseau ETHERNET. Le réseau est conçu pour désservir vingt personnes dans un même site. Une longueur maximale de 200 m pour les bus est suffisante.

Comme le montre la figure 2.2, les trois niveaux habituels de protocoles sont présents :

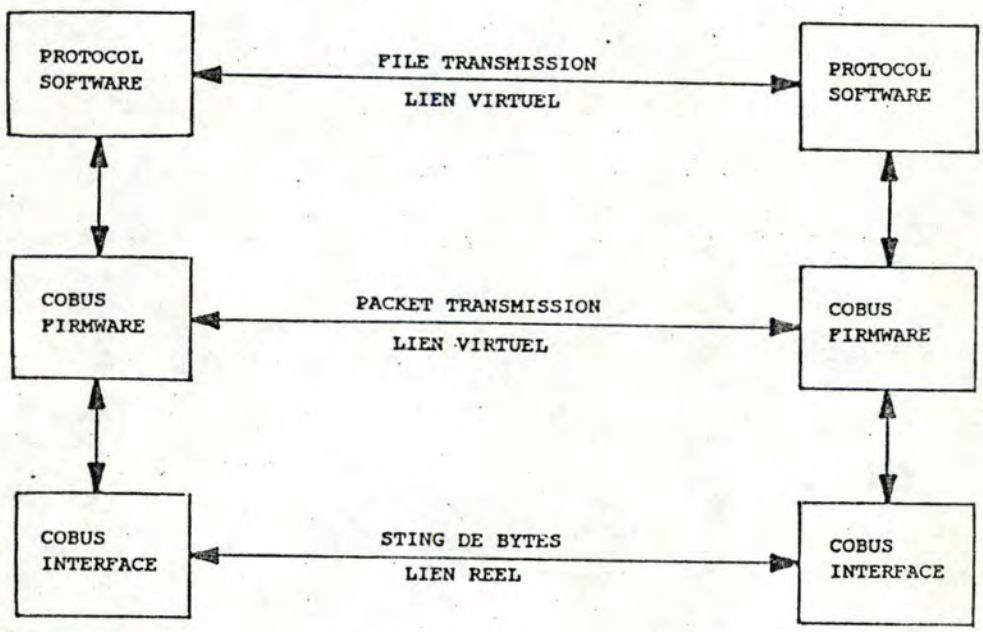


Fig. 2.2. Niveaux de protocoles

Au niveau le plus bas, les paquets sont envoyés d'une manière aléatoire aux différentes unités mais un mécanisme évite les collisions entre les paquets et détecte les interférences. La vitesse de transmission est de 375 kbits par seconde.

Le second niveau du protocole garantit la cohérence du paquet transféré.

Enfin, le troisième niveau sert aux transferts de fichiers. Les trames sont stockées dans une PROM et transmises par le microprocesseur Z80.

2.7. Caractèristiques du terminal VISUAL 200

Le terminal VISUAL 200 [VISUAL TECHNOLOGY INCORPORATED 80] a une double fonction :

- il envoie à l'ordinateur via un émetteur de l'information introduite au clavier
- il peut afficher à l'écran les réponses provenant de l'ordinateur via un récepteur

La figure 2.3. illustre les fonctions du terminal VISUAL 200.

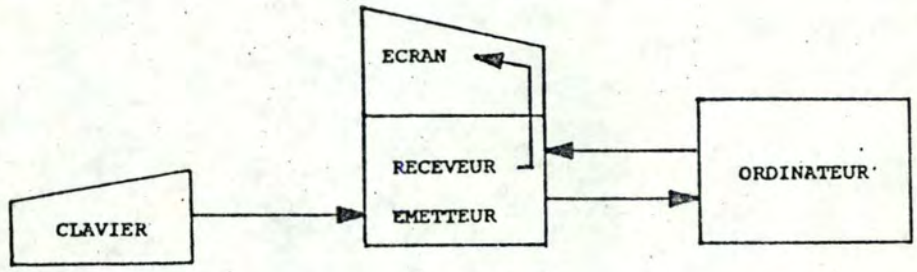


Fig. 2.3. Fonction du terminal

Il contient beaucoup d'articles standards : matrice de 7 * 9 points, un éditeur de lignes et de caractères, des tabulations, un curseur adressable, des "switchs" de sélection ... il est doté également de plusieurs modes : DEC VT 52, ADDS 520, LSIADM 3A, HARZELINE 1500 et est équipé d'une configuration européenne.

Le clavier du VT 200 se compose de 94 touches (numériques, alphabétiques, et 13 touches de fonctions) et de cinq indicateurs visuels (mode local ou normal, majuscule ou minuscule et les trois autres sont programmables par l'utilisateur)

L'écran mesure 12 pouces en diagonale et peut afficher 24 lignes de 80 caractères ou 12 lignes de 132 caractères. L'écran est alphanumérique avec des possibilités graphiques. Quand le clavier est absent, le terminal est aussi fonctionnel uniquement en mode lecture.

Chapitre 3 :

ANALYSE FONCTIONNELLE

Chapitre 3:

ANALYSE FONCTIONNELLE

3.1. Introduction

Dans le cadre de notre mémoire, nous avons élaboré un formateur, formateur que nous avons implémenté lors de notre stage effectué au laboratoire de micro-informatique de l'Ecole Polytechnique Federale de Lausanne (E.P.F.L.). Les caractéristiques générales ont été définies en fonction de l'analyse de l'existant décrite au chapitre 1. Ce formateur se situe dans la catégorie des "vrais" formateurs.

Dans ce chapitre, nous présenterons l'analyse fonctionnelle. L'analyse fonctionnelle consiste à définir le " Que Faire " sans s'occuper du " Comment Faire ".

Cette analyse fonctionnelle comportera trois parties :

- la première partie consiste en une description des différentes contraintes qui nous ont été données à l'E.P.F.L.
- la deuxième partie présente le contexte d'utilisation du formateur
- la troisième partie présente les fonctions du formateur

3.2. Description des contraintes de l'E.P.F.L.

Dans le cadre de notre mémoire, il nous a été demandé de réaliser un formateur de document. Nous avons réalisé ce formateur à l'E.P.F.L où certaines contraintes matérielles nous étaient imposées :

- Le formateur devait être implémenté sur Vax, mais transportable plus tard sur Smaky.

- L'output de notre formateur devait être un fichier binaire dans un format standard. Ce fichier comprend toutes les indications nécessaires pour l'impression d'un document. Le format est le standard : "FORMATTED DOCUMENT DESCRIPTION" (en abrégé FDD), il est décrit dans l'annexe C.

3.3. Contexte d'utilisation du formateur

3.3.1. Rappel

Le principe utilisé lors de la description des traitements est celui de décomposition hiérarchique. Schéma à la figure 3.1.

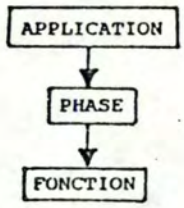


Fig. 3. 1. Hiérarchie des traitements

Expliquons quelque peu les différents concepts de cette hiérarchie.

L'"application" regroupe un ensemble de phases relatives à un flux homogène d'informations ayant une permanence dans l'organisation. Une application se compose de phases.

La "phase" est un traitement manuel ou automatisable possédant une unité spatio-temporelle. Cette unité d'exécution implique que la phase soit entièrement exécutée dans une cellule d'activité. Une cellule d'activité est un centre d'activité homogène dans le temps et dans l'espace, doté de ressources humaines et/ou matérielles et pourvue de règles de comportement nécessaires à son fonctionnement. Une phase se compose de fonctions.

La "fonction" représente des traitements considérés comme élémentaires. Elle est associée à un objectif et à un comportement considérés comme élémentaires dans l'organisation.

3.3.2. Contexte dans lequel le formateur est placé

3.3.2.1. Introduction

Dans la description du contexte, nous allons placer le formateur dans son contexte d'utilisation.

Le formateur réalisé est une phase d'un système de traitement de texte. L'ensemble sera appelé "application de traitement de texte".

L'application sera constituée de trois phases :

- la "phase d'édition" comprend l'encodage d'un texte.
- la "phase de formatage" a comme but de formater un document (*).
- la "phase d'impression" sera constituée d'un certain nombre de manipulations.

La figure 3.2. schématise cette découpe.

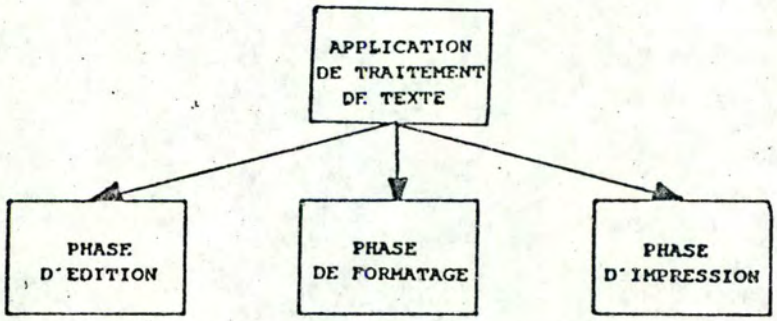


Fig. 3.2. Système de traitement de texte

3.3.2.2. Description de la phase d'édition

La phase d'édition consiste à réaliser l'encodage du texte source comprenant des commandes pour le formateur.

Les "vrais" formateurs (comme défini dans le premier chapitre) sont réalisés indépendamment de l'édition.

Comme "FTTL" se situe dans la catégorie des "vrais" formateurs, nous n'avons pas développé un système d'édition spécifique.

3.3.2.3. Description de la phase de formatage

La phase de formatage a comme but, à partir d'un texte source au kilomètre, de construire un fichier binaire dans le format FDD. Comme il a déjà été signalé, le fichier binaire dans le format FDD comprend toutes les indications nécessaires pour l'impression. Ceci est la phase qui nous a été demandée de concevoir et d'implémenter, la réalisation de cette phase correspond donc à la réalisation du formateur proprement dit.

Le formateur sera un sytème batch.

Le formateur disposera d'un fichier qui contiendra le texte au kilomètre et de fichiers qui contiendront des indications quant à la taille des caractères disponibles. (A chaque jeu de caractères, un jeu est défini pour une fonte et une police , on associera un fichier ; pour plus de détail, se référer à la section 4.4.5. page 4.21). Chaque fichier comprendra pour chacun des 128 caractères ASCII sa hauteur et sa largeur.

Le formateur créera un fichier binaire FDD (comme dit ci-dessus) et un fichier qui contiendra les erreurs que l'utilisateur aura faites dans le langage du formateur (Le fichier sera créé dans tous les cas, si l'utilisateur n'a pas commis d'erreur le fichier sera vide)

La figure 3.3 illustre la phase de formatage, avec les fichiers en entrée et les fichiers en sortie

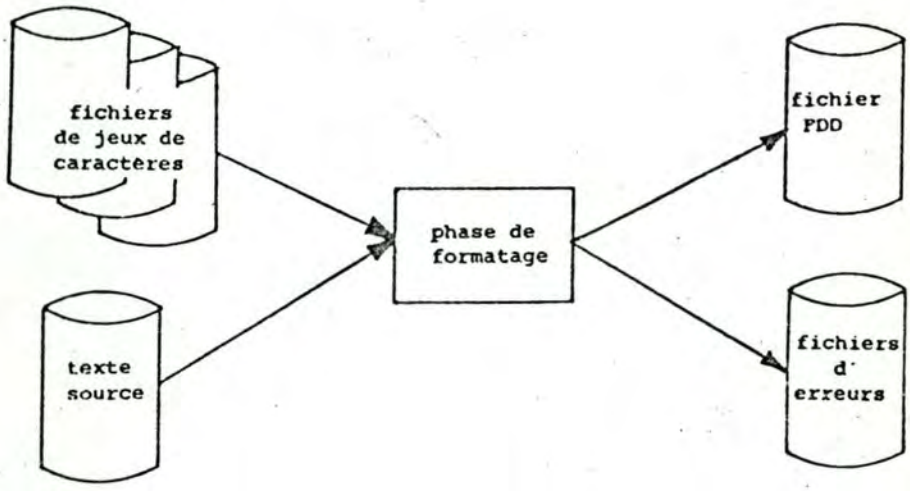


Fig. 3.3. Schéma de la phase de formatage

Les différentes fonctions du formateur sont décrites à la section 3.4.

3.3.2.4. Description de la phase d'impression

Cette phase a comme but, à partir du fichier binaire en format FDD, réalisé lors de la phase de formatage, de réaliser les transferts et les chargements nécessaires afin de pouvoir effectuer l'impression du document à l'aide de l'imprimante laser.

Pour notre part, nous avons réalisé le programme de décodage du format binaire en format "S" (le programme, les spécifications et le mode d'emploi ainsi que la description des différents formats se trouvent dans l'annexe D).

La phase de formatage est réalisée sur le VAX (tout le matériel a été décrit dans le chapitre 2) tandis que l'unité d'impression est connectée au réseau COBUS. Il faut donc transférer le fichier sur COBUS. Pour réaliser l'opération de transfert, il faut encoder le fichier binaire en format "S". Une fois le fichier stocké sur le réseau COBUS, il faut encore le transformer en format "SM" pour qu'il puisse être traité par l'interface de l'imprimante. Lorsque le fichier est encodé, il suffit de le charger et de charger les fichiers de jeux de caractères ; finalement, il faut lancer la commande d'impression pour avoir le document formaté et imprimé. La figure 3.4 (page suivante) illustre ces transferts et chargements.

Cette phase actuellement est essentiellement manuelle. Un interface gérant tous les transferts et chargements est en cours de réalisation (à l'E.P.F.L. par P. Fäh) et permettra l'envoi d'un document à imprimer à partir du Vax sur l'imprimante laser, en passant au travers des différents réseaux.

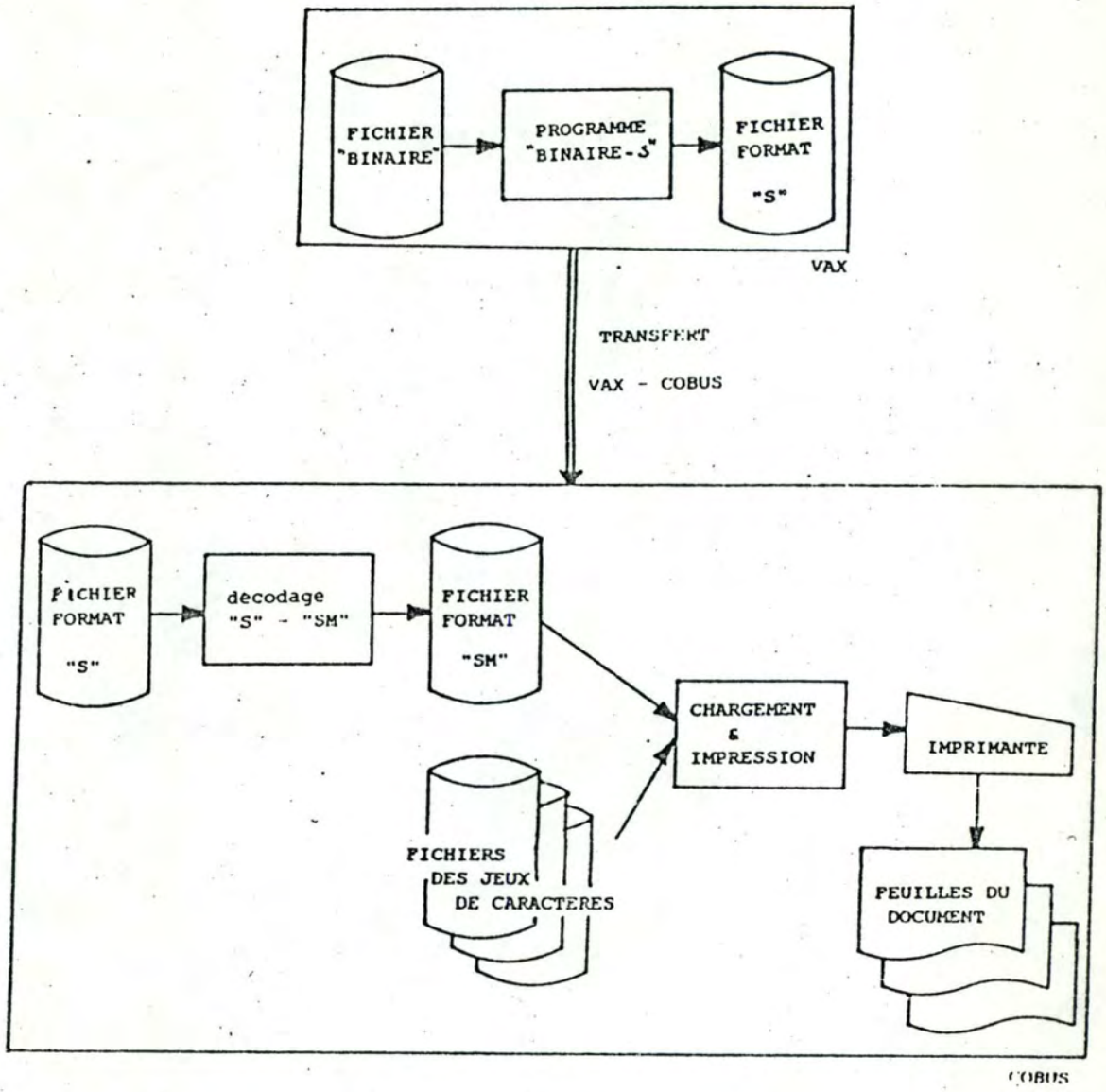


Fig. 3.4. Chaîne d'impression

3.4. Fonctions du formateur

3.4.1. Introduction

Cette section reprend les spécifications que nous nous sommes fixées afin de réaliser un outil performant.

Les éléments les plus importants du formateur à réaliser sont :

- la facilité d'utilisation (au niveau du langage)
- le grand nombre de commandes et de possibilités offertes
- l'obtention d'une mise en page "optimale"

Ces différents éléments sont développés dans les sections suivantes.

3.4.2. La facilité d'utilisation

Les commandes constituant le langage de formatage devront être représentatives de leur effet et devront offrir à l'utilisateur une certaine liberté dans leur emploi.

C'est ainsi que nous aurons :

- le français comme langue utilisée pour les commandes
- la possiblité d'utiliser indifféremment les caractères majuscules et/ou minuscules pour les commandes
- la possibilité d'introduire des commandes sous une forme "réduite", c'est-à-dire avec un minimum de caractères significatifs
- les commandes insérées aléatoirement dans le texte (même type d'insertion que celui utilisé par le formateur FORMAT décrit dans le chapitre 1)
- le backslash comme unique caractère permettant de distinguer une commande du texte
- un système de traitement de macros indépendant du formateur (le système correspond à un pré-processeur)

Un système de gestion des erreurs permettra à l'utilisateur de repérer facilement une erreur dans la syntaxe d'une commande.

L'ensemble des commandes ainsi que leurs spécifications se trouvent dans le Manuel Utilisateur.

3.4.3. Les fonctions de formatage

Dans l'analyse des systèmes existants, nous avons constaté que la dernière génération de formateurs (TEX, SCRIBE) était basée essentiellement sur des notions de blocs, d'environnements. Un bloc, environnement est caractérisé par un ensemble d'attributs de formatage.

Suite à cette étude, nous avons effectué une étude des éléments pouvant constituer un document. Un problème est apparu sur la notion de document, car il nous était impossible de cerner le contenu de quelque chose de non précisé. Le formateur SCRIBE formate en fonction d'un type de document : lettre, article, revue, ... ; nous nous sommes donc limités à traiter des documents bien spécifiques : ceux de type "livre" (*). Dès lors, nous associerons toujours par la suite la notion de document à celle de "livre".

Nous avons établi qu'un document était constitué essentiellement de chapitres, d'entêtes de chapitre, de paragraphes, de listes, de figures avec un texte, de notes en bas de page ; outre ces éléments constituant le corps du document, un certain nombre d'autres éléments permettent l'identificatio de ce dernier : le titre du document, le nom de ou des auteur(s), la date (de parution ou de rédaction) et un texte (soit un résumé du document, soit une introduction, soit une préface, ...). L'ensemble de ces elements constituant la structure seront appelés "élément logique" (*).

Les types d'éléments logiques que nous définirons seront : le titre (*), l'auteur (*), la date (*), le résumé (*), le chapitre (*), l'entête (*), le paragraphe (*), la liste (*), la figure (*), la note bas de page (*).

A chaque type d'élément logique sera associé un ensemble de paramètres de présentation (*). Ces paramètres seront : le centrage (*), la marge gauche (*), la marge droite (*), la fonte (*), le soulignement (*), la police (*), le décalage (*), l'implémentation (*), l'interligne (*), la hauteur (*) et la largeur (*) d'une figure. Une valeur sera associée à priori pour chacun de ses paramètres. Elle pourra être modifiée soit pour tous les éléments logiques d'un même type, soit pour un élément logique, soit pour une partie d'un élément logique.

Un certain nombre de fonctions seront définies en plus de celles de formatage des éléments logiques. C'est ainsi que le formateur devra permettre :

- la création d'une table des matières (*) sur demande (table qui contiendra les entêtes de chapitre ainsi que pour chaque entête le numéro de la page contenant l'entête)

- la création d'un index (*) sur demande (index qui contiendra des mots annotés ainsi que pour chaque mot, le numéro de la page contenant le mot)

- la pagination (*) en bas de page ou en haut de page sur demande

- la numérotation automatique des entêtes de chapitre, des figures et des notes bas de page

- la mise en page en simple ou en double colonne sur demande (le colonnage (*))

- la possibilité de modifier la grandeur des marges supérieures (*) ou/et des marges inférieures (*)

- la possibilité de modifier le nombre de millimètres entre deux éléments logiques (interentité (*))

- la possibilité de ne traiter qu'une partie du document (impression réduite (*)) et de spécifier la première et la dernière page du document à imprimer

- la possibilité pour l'utilisateur d'intialiser le numéro des figures, des notes bas de page, de l'entête de chapitre de niveau 1

- la possibilité de formater pour n'importe quel type de feuille (*), le type de feuille déterminant une largeur et une hauteur

- la possibilité d'obtenir les éléments d'identification centrés sur une feuille indépendante et non numérotée (une page (*))

3.4.4. L'obtention de la mise en page "optimale"

3.4.4.1. Introduction

Afin d'obtenir une mise en page optimale, nous avons utilisé le concept de "colle" décrit par Knuth [Knuth 79].

La notion de "colle" consiste à faire varier l'espace entre les éléments logiques qui seront imprimés sur la page.

De plus, deux contraintes seront prises quant à la mise en page des éléments logiques de type entête, figure, note bas de page et pour la réalisation du double colonnage.

3.4.4.2. Notion de "colle"

La mise en page optimale sera réalisée afin d'éviter qu'un paragraphe (par exemple) n'ait une première ligne sur une page et le reste sur l'autre. La mise en page optimale sera réalisée grâce au concept de "colle".

Une page est formée de N entité(s) séparée(s) par un espace d'une dimension variant entre une valeur minimale et une valeur maximale (N > 0).

Au niveau du simple colonnage, on peut considérer les trois cas suivants :

- premier cas : une entité peut prendre place sur la page courante et ce en prenant une valeur pour l'écart entre entités comprise entre une valeur minimale et une valeur moyenne, valeur prédéfinie. Sur la figure 3.5, le schéma de gauche présente les entités avec un écart moyen et celui de droite avec l'écart optimal compris entre l'écart minimal et l'écart moyen.

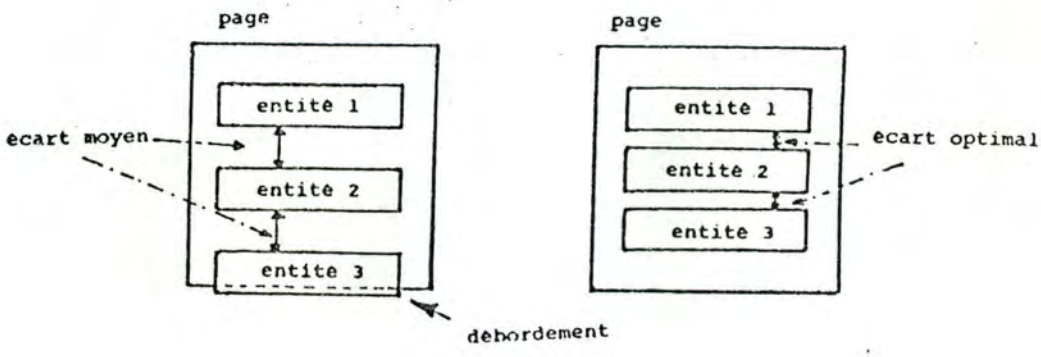


Fig. 3.5. Mise en page. Cas de reserrement

- deuxième cas : une entité peut prendre place sur la page suivante et ce en prenant une valeur pour l'écart entre entités compris entre une valeur moyenne, valeur pré-définie, et la valeur maximale. Sur la figure 3.6, le schéma de gauche présente les entités avec un écart moyen et celui de droite avec un écart optimal compris entre l'écart moyen et l'écart maximal.

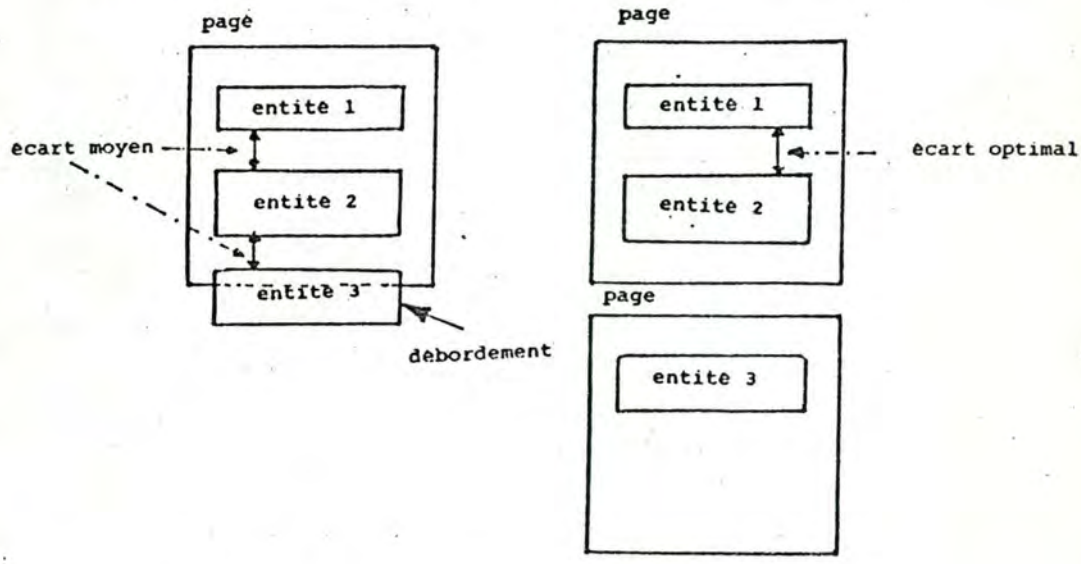
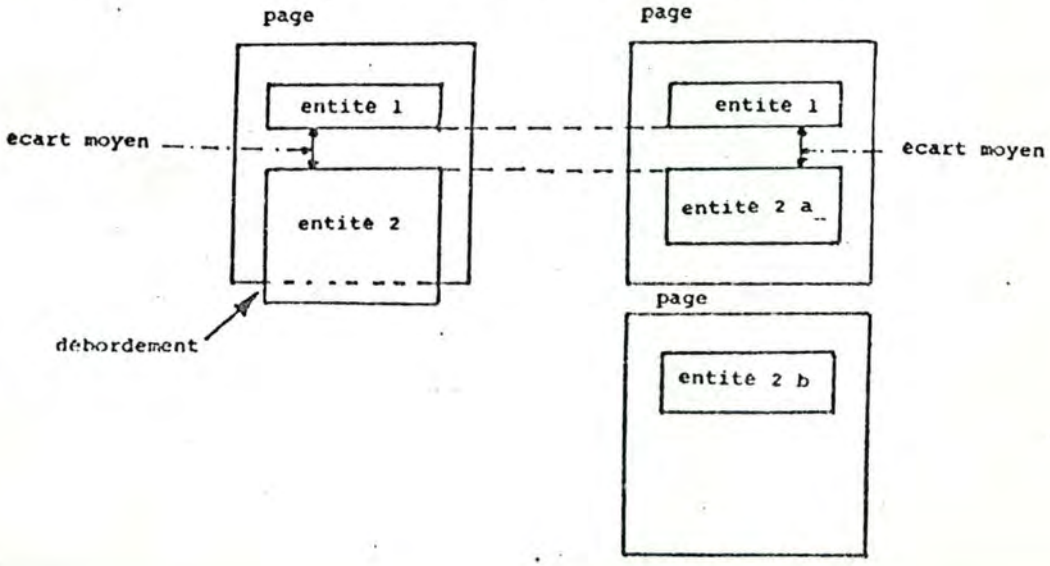


Fig. 3.6. Mise en page. Cas d'élargissement

- troisième cas : une entité ne peut prendre place sur la page courante, l'écart devrait être inférieur à la valeur minimale et ne pourra non plus être placé totalement sur la page suivante, l'écart serait dans ce cas supérieur à la valeur maximale entre entités. Dans ce cas, l'entité en question sera "cassée". Sur la figure 3.7, le shéma de gauche présente les entités avec un écart moyen, la valeur de cet écart sera conservée entre l'entité 1 (sur le schéma de gauche) et l'entité 2a (sur le schéma de droite). L'entité 2 (schéma de gauche) sera "cassée" en deux entités (schéma de droite) : 2a qui se placera sur la page courante et 2b sur la page suivante.

Fig. 3.7. Mise en page. Cas de cassure



3.4.4.3. Contraintes de mise en page

Un élément logique de type "figure" ne pourra être cassé et devra être impérativement suivi de son texte et cela, sur la même page.

Un élément logique de type note bas de page devra impérativement être sur la même page que sa référence.

Un élément logique de type entête ne pourra se trouver en bas de page sans être suivi d'au moins une partie d'un élément logique.

Le problème du double colonnage reposera le problème de la mise en page décrit à la section 3.4.4.2. avec le choix en plus de passer à la page ou non.

Chapitre 4 :

ANALYSE ORGANIQUE

Chapitre 4:

ANALYSE ORGANIQUE

4.1. Introduction

Les deux premières étapes du projet, à savoir l'analyse de l'existant et l'analyse fonctionnelle constituent les supports indispensables à l'analyse organique.

Méthodologiquement, l'analyse organique comporte trois phases :

- la définition d'une architecture c'est-à-dire un ensemble structuré de programmes travaillant sur un ensemble de fichiers
- l'écriture des programmes dans un langage choisi
- la réalisation de tests validant ces programmes

L'analyse organique détaillera ces trois phases en reprenant successivement :

- le choix du langage de programmation
- la description physique de la phase de formatage
- la structure des fichiers
- la définition d'une architecture physique
- la spécification des modules
- les méthodes de tests
- la description des données

4.2. Choix du langage

Le choix du langage de programmation succède généralement à la découpe physique. Nous le présentons en premier lieu car il a motivé le choix de notre architecture.

Initialement, le formateur devait être implémenté en MODULA à la demande des assistants de LAUSANNE, mais le laboratoire de micro-informatique n'a reçu le compilateur MODULA de ZURICH que fin janvier 83 ; pour cet raison, nous avons choisi un langage de haut niveau le plus compatible avec MODULA : PASCAL.

D'autres facteurs ont influencé notre choix :

- quand on invoque PASCAL, on pense immédiatement à un langage de programmation structuré

- PASCAL est supposé être suffisamment standard pour être exécuté sur plusieurs machines.

  Il y a certainement plus d'uniformité parmi les implémentations PASCAL que parmi les implémentations des autres langages ; cependant, il subsiste encore des différences entre les versions de PASCAL car il est très difficile de réaliser un compilateur standard. Des recherches ont été faites dans ce sens par R. CICHELLE de l'institut de recherche ANPA'S [LOGITECK 82] .

- PASCAL permet d'utiliser également des procédures et/ou des fonctions qui s'invoquent elles-mêmes, c'est-à-dire que PASCAL admet la récursivité ; cette possibilité offre une approche élégante et efficace pour résoudre les problèmes du traitement de texte qui n'est pas disponibles dans tous les autres langages de haut niveau.

4.3. Description physique de la phase de formatage

4.3.1. Introduction

Comme il a été signalé dans le chapitre 3, la phase de formatage a comme but, à partir d'un texte au kilomètre, de construire un fichier binaire dans le format "FDD", fichier qui comprendra toutes les indications nécessaires pour l'impression du texte source.

Cette phase se découpe en 4 étapes successives :

- la première étape définit la valeur des paramètres de présentation (*)
- la deuxième étape construit l'arbre de structure (*)
- la troisième étape construit l'arbre de formatage (*)
- la quatrième étape construit le fichier binaire dans le format "FDD"

La figure 4.1 illustre les 4 étapes de la phase de formatage. Les sections suivantes décrivent ces différentes étapes.

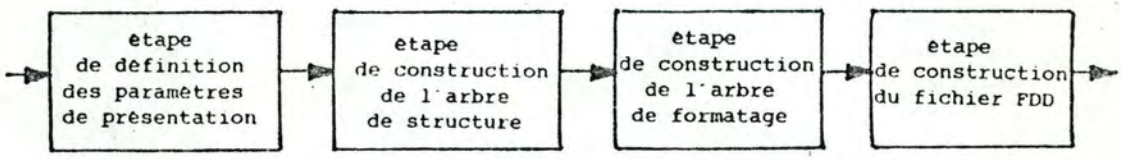


Fig. 4.1. Illustration des étapes de la phase de formatage

4.3.2. Définition des paramètres de présentation

L'étape de définition des paramètres de présentation associera pour chaque élément logique des valeurs à ses paramètres de présentation.

Par exemple, un élément logique de type paragraphe aura :

- la grandeur de sa marge gauche à "0"
- la grandeur de sa marge droite à "0"
- le type de fonte sera le type ‹normal›
- le type de police sera ‹11›
- le type de soulignement sera ‹rien›

De plus, il faudra définir des valeurs pour les paramètres de présentation concernant l'ensemble du document : pas de table des matières, simple colonne, pas d'index, ...

Afin de permettre à l'utilisateur d'avoir un formatage "à la carte", un certain nombre de commandes se trouveront en entête du texte à formater et permettront de modifier les valeurs pré-définies.

4.3.3. Etape de construction de l'arbre de structure

4.3.3.1. Introduction

L'étape de construction de l'arbre de structure aura comme but, à partir d'un texte au kilomètre de construire un arbre qui sera la représentation structurale du document. L'arbre construit sera une représentation LOGIQUE du document.

Un fichier contenant les erreurs que l'utilisateur aura pu faire dans le langage de formatage sera également disponible.

La figure 4.2 illustre cette étape.

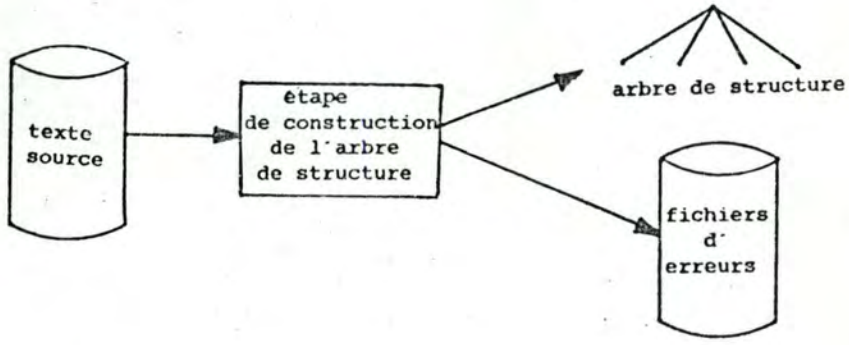


Fig 4.2. Etape de construction de l'arbre de structure

Cette section comprendra :

- une description du contenu de l'arbre de structure
- une figure illustrant la représentation d'un arbre de structure (figure 4.6, page 4.12), cette représentation correspondra à un texte (figure 4.5, page 4.10)

4.3.3.2. Description du contenu de l'arbre de structure

L'arbre comprendra des éléments correspondant à la structure d'un document.
Ainsi, nous pourrons rencontrer les types d'éléments suivants :

- un titre de document
- un auteur
- une date de création
- un résumé
- un chapitre et son niveau
- un entête de chapitre
- un paragraphe
- une liste de plusieurs éléments
- une note bas de page
- une figure

En premier lieu, à chacun des éléments logiques un certain nombre de paramètres de présentation seront associés pour le formatage.
Ainsi, nous aurons comme paramètres de présentation :

- l'indicateur de centrage
- la grandeur de la marge gauche
- la grandeur de la marge droite
- la grandeur de l'interligne
- le type de soulignement désiré
- le type de fonte désiré
- le type de police désiré

Certains éléments de la structure devront comporter des paramètres complémentaires.
Ainsi, nous aurons en plus :

- pour un chapitre, son niveau
- pour une liste, son type d'implémentation et son décalage
- pour un paragraphe, la grandeur du décalage de sa première ligne
- pour une figure, sa hauteur et sa largeur

Le premier élément de la structure d'arbre, c'est-à-dire celui de type "document", devra contenir des paramètres nécessaires pour l'étape de construction de l'arbre de formatage.
C'est ainsi que nous aurons :

- le type de la feuille pour l'impression (déterminant la longueur et la largeur utile de la page)

- le paramètre indiquant que le document devra ou non comporter une table des matières, ainsi que le nombre de niveaux d'entête qu'il faudra y indiquer

- le paramètre indiquant que le document devra être paginé ou non, le numéro de page devant se trouver en haut ou en bas de page

- le paramètre indiquant que le document devra être en double colonne ou en simple

- le paramètre indiquant que le document devra ou non comporter un index

- le paramètre indiquant que les éléments d'identification du document devront se trouver centrés sur une page indépendante

- le paramètre indiquant les pages qui devront être imprimées

- le paramètre indiquant le nombre moyen de millimètres entre deux éléments logiques

- les paramètres indiquant la grandeur de la marge inférieure et de la marge supérieure

- les paramètres pour la présentation de la table des matières (fonte, police, ...)

En deuxième lieu, à chacun des éléments logiques, on devra associer des pointeurs afin de réaliser les liens entre les éléments logiques.
Chaque élément logique comportera 4 pointeurs (au maximum et suivant les nécessités) :

- les deux premiers contiennent l'adresse de l'élément logique suivant, élément de type différent, c'est ce que nous désignons "successeur"

- le troisième contient l'adresse de l'élément logique suivant, élément de même type, c'est ce que nous désignons "suivant"

- le quatrième contient l'adresse vers le premier mot

En troisième lieu, à chacun des éléments logiques, sauf l'élément logique de type document et l'élément logique de type chapitre, on devra associer les caractères du texte source. ces caractères seront regroupés en mots (*). La suite des mots sera associée à l'élément logique de l'arbre de structure les contenant. A chaque mot devra être associé un certain nombre de paramètres : le type de soulignement, le type de fonte, le type de police ainsi qu'un indicateur de saut de ligne. La figure 4.3. illustre les différents paramètres associés à un mot.

STRING	tableau de 24 caractères
TYPE DE POLICE	10,11,14 ou 20
TYPE DE FONTE	gras,italique,normal
TYPE DE SOULIGNEMENT	continu,discontinu,supérieur, rien
INDICATEUR DE SAUT DE LIGNE	booléen
NOMBRE DE CARACTERES	entier
POINTEUR SECONDAIRE	pointeur vers le mot suivant

Fig. 4.3. Paramètres associés à un "mot"

La figure 4.4. (page suivante) illustre le contenu d'un élément logique de l'arbre de structure.

Remarque : si chaque élément logique comprenait le même nombre de paramètres de présentation, on perdrait de la place mémoire étant donné que tous les paramètres ne sont pas nécessaires pour chaque élément logique. Le langage PASCAL permet d'éviter cette perte de place par l'utilisation du "case booléen"

TYPE D'ELEMENT LOGIQUE	titre,auteur,date,resume, chapitre,paragraphe,figure,entete, note bas de page
INDICATEUR D'INDEX	booleen
INDICATEUR D'IMPRESSION REDUITE	booleen
PREMIERE PAGE A IMPRIMER	entier
DERNIERE PAGE A IMPRIMER	entier
NUMERO DE LA PREMIERE PAGE	entier
INDICATEUR DE "SEUL SUR PAGE"	booleen
INDICATEUR DE PAGINATION	booleen
INDICATEUR DE COLONNAGE	booleen
INDICATEUR DE TABLE DES MATIERES	booleen
INDICATEUR DE CENTRAGE	booleen
TYPE DE FEUILLE	A4
LARGEUR D'UNE FEUILLE	entier
LONGUEUR D'UNE FEUILLE	entier
INTERENTITE	nombre de millimètres
MARGE GAUCHE	nombre de millimètres
MARGE DROITE	nombre de millimètres
MARGE SUPERIEURE	nombre de millimètres
MARGE INFERIEURE	nombre de millimètres
LARGEUR D'UNE FIGURE	nombre de millimètres
HAUTEUR D'UNE FIGURE	nombre de millimètres
NIVEAU D'UN CHAPITRE	entier compris entre 1 et 4
DECALAGE	nombre de millimètres
TYPE DE FONTE	gras,normal,italique
TYPE DE POLICE	10,11,14,20
TYPE DE SOULIGNEMENT	continu,discontinu,superieur,rien
TYPE DE LISTE	point,tiret,etoile
INTERLIGNE	nombre de millimètres
INDICATEUR DE SAUT DE PAGE	booleen
INDICATEUR DE SAUT DE LIGNE	booleen
POINTEUR SUCCESSEUR 1	pointeur vers l'élément suivant de type différent
POINTEUR SUCCESSEUR 2	pointeur vers l'element suivant de type different
POINTEUR SUIVANT	pointeur vers l'élément suivant de même type
POINTEUR SECONDAIRE	pointeur vers le mot

Fig. 4.4. Ensemble des attributs associés à un élément logique

4.3.3.3. Exemple de représentation d'un arbre de structure

4.3.3.3.1. Texte

La figure 4.5. (ci-dessous) présente un exemple de texte formaté, la figure 4.6. (page 4.12) présentera la représentation du texte sous la forme de l'arbre de structure et la figure 4.10. (page 4.17), sous la forme de l'arbre de formatage.

EXEMPLE DE REPRESENTATION

1. Ceci est un chapitre

1.1. Ceci est un sous-chapitre

Ceci est un paragraphe décalé de 10 mm

Ceci est une liste :

- premier élément
- second élément

2. Ceci est un chapitre

Fig. 4.5. Exemple de texte

Remarque :

Les indications contenues dans l'arbre de structure et l'arbre de formatage représentant cet exemple ne seront pas exactement représentatives du contenu de la figure 4.5. vu que l'on ne peut disposer des mêmes types de soulignement, fonte et police sur l'imprimante Laser Canon Lbp-10 et sur l'imprimante Sanders.

4.3.3.3.2. Représentation de l'arbre de structure

La figure 4.6 (page suivante) illustre la représentation d'un arbre de structure, il représente le texte de la figure 4.5.. Deux types d'éléments constituent cet arbre, les records contenant les informations des éléments logiques, où sont données les attributs ainsi que leurs valeurs, et les records de type mot. Pour ces derniers, nous utiliserons un code pour spécifier les attributs (0 = nombre de caractères, 1 = string, 2 = type de soulignement, 3 = type de police, 4 = type de fonte, 5 = indicateur de saut de ligne, 6 = pointeur vers le mot suivant)

TYPE D'ELEMENT LOGIQUE	document
INDICATEUR D'INDEX	non
INDICATEUR D'IMPRESSION REDUITE	non
PREMIERE PAGE A IMPRIMER	0
DERNIERE PAGE A IMPRIMER	0
NUMERO DE LA PREMIERE PAGE	1
INDICATEUR DE PAGINATION	non
INDICATEUR DE COLONNAGE	non
INDICATEUR DE TABLE DES MATIERES	non
LARGEUR D'UNE FEUILLE	210
HAUTEUR D'UNE FEUILLE	297
INDICATEUR DE SEUL SUR PAGE	non
MARGE SUPERIEURE	30
MARGE INFERIEURE	30
POINTEUR SUIVANT	

0	8
1	Exemple
2	rien
3	14
4	gras
5	non
6	

0	3
1	de
2	rien
3	14
4	gras
5	non
6	

0	13
1	representation
2	rien
3	14
4	gras
5	non
6	nil

TYPE D'ELEMENT LOGIQUE	titre
INDICATEUR DE CENTRAGE	oui
MARGE GAUCHE	30
MARGE DROITE	30
TYPE DE POLICE	14
TYPE DE FONTE	gras
TYPE DE SOULIGNEMENT	rien
INTERLIGNE	5
INDICATEUR DE SAUT DE LIGNE	non
INDICATEUR DE SAUT DE PAGE	non
POINTEUR SUCCESSEUR 1	
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	nil
POINTEUR SECONDAIRE	

TYPE D'ELEMENT LOGIQUE	chapitre
NIVEAU D'UN CHAPITRE	1
POINTEUR SUCCESSEUR 1	
POINTEUR SUCCESSEUR 2	
POINTEUR SUIVANT	

TYPE D'ELEMENT LOGIQUE	chapitre
NIVEAU D'UN CHAPITRE	1
POINTEUR SUCCESSEUR 1	
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	nil

TYPE D'ELEMENT LOGIQUE	entete
INDICATEUR DE CENTRAGE	non
MARGE GAUCHE	10
MARGE DROITE	10
TYPE DE POLICE	11
TYPE DE FONTE	gras
TYPE DE SOULIGNEMENT	continu
INTERLIGNE	3
NIVEAU DU CHAPITRE	1
INDICATEUR DE SAUT DE LIGNE	non
INDICATEUR DE SAUT DE PAGE	non
POINTEUR SUCCESSEUR 1	nil
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	nil
POINTEUR SECONDAIRE	

0	5
1	Ceci
2	continu
3	11
4	gras
5	non
6	

0	4
1	est
2	continu
3	11
4	gras
5	non
6	

0	3
1	un
2	continu
3	11
4	gras
5	non
6	

0	8
1	chapitre
2	continu
3	11
4	gras
5	non
6	Nil

TYPE D'ELEMENT LOGIQUE	chapitre
NIVEAU D'UN CHAPITRE	2
POINTEUR SUCCESSEUR 1	
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	nil

TYPE D'ELEMENT LOGIQUE	entete
INDICATEUR DE CENTRAGE	non
MARGE GAUCHE	10
MARGE DROITE	10
TYPE DE POLICE	11
TYPE DE FONTE	normal
TYPE DE SOULIGNEMENT	continu
INTERLIGNE	3
NIVEAU DU CHAPITRE	2
INDICATEUR DE SAUT DE LIGNE	non
INDICATEUR DE SAUT DE PAGE	non
POINTEUR SUCCESSEUR 1	nil
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	nil
POINTEUR SECONDAIRE	

0	5
1	Ceci
2	rien
3	11
4	normal
5	non
6	

0	4
1	est
2	rien
3	11
4	normal
5	non
6	

0	3
1	un
2	rien
3	11
4	normal
5	non
6	

0	11
1	paragraphe
2	rien
3	11
4	normal
5	non
6	

0	7
1	decale
2	rien
3	11
4	normal
5	non
6	

0	3
1	de
2	rien
3	11
4	normal
5	non
6	

0	3
1	10
2	rien
3	11
4	normal
5	non
6	

0	2
1	mm
2	rien
3	11
4	normal
5	non
6	nil

TYPE D'ELEMENT LOGIQUE	paragraphe
INDICATEUR DE CENTRAGE	non
MARGE GAUCHE	10
MARGE DROITE	10
DECALAGE	10
TYPE DE POLICE	11
TYPE DE FONTE	normal
TYPE DE SOULIGNEMENT	rien
INTERLIGNE	1
INDICATEUR DE SAUT DE LIGNE	non
INDICATEUR DE SAUT DE PAGE	non
POINTEUR SUCCESSEUR 1	nil
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	
POINTEUR SECONDAIRE	

0	5
1	Ceci
2	rien
3	11
4	normal
5	non
6	

0	4
1	est
2	rien
3	11
4	normal
5	non
6	

0	3
1	un
2	continu
3	11
4	gras
5	non
6	

0	8
1	chapitre
2	continu
3	11
4	gras
5	non
6	Nil

0	5
1	Ceci
2	rien
3	11
4	normal
5	non
6	

0	4
1	est
2	rien
3	11
4	normal
5	non
6	

0	3
1	un
2	rien
3	11
4	normal
5	non
6	

0	8
1	chapitre
2	rien
3	11
4	normal
5	non
6	Nil

0	5
1	Ceci
2	rien
3	11
4	normal
5	non
6	

0	4
1	est
2	rien
3	11
4	normal
5	non
6	

0	3
1	un
2	rien
3	11
4	normal
5	non
6	

0	8
1	chapitre
2	rien
3	11
4	normal
5	non
6	Nil

TYPE D'ELEMENT LOGIQUE	liste
INDICATEUR DE CENTRAGE	non
MARGE GAUCHE	10
MARGE DROITE	10
DECALAGE	10
TYPE DE POLICE	11
TYPE DE FONTE	normal
TYPE DE SOULIGNEMENT	rien
INTERLIGNE	1
TYPE DE LISTE	tiret
INDICATEUR DE SAUT DE LIGNE	non
INDICATEUR DE SAUT DE PAGE	non
POINTEUR SUCCESSEUR 1	nil
POINTEUR SUCCESSEUR 2	nil
POINTEUR SUIVANT	nil
POINTEUR SECONDAIRE	

0	2
1	-
2	rien
3	11
4	normal
5	non
6	

0	8
1	premier
2	rien
3	11
4	normal
5	non
6	

0	8
1	element
2	rien
3	11
4	normal
5	non
6	

0	2
1	-
2	rien
3	11
4	normal
5	oui
6	

0	7
1	second
2	rien
3	11
4	normal
5	non
6	

0	8
1	element
2	rien
3	11
4	normal
5	non
6	NIL

0	4
1	une
2	rien
3	11
4	normal
5	non
6	

0	7
1	liste :
2	rien
3	11
4	normal
5	non
6	NIL

4.3.4. Etape de construction de l'arbre de formatage

4.3.4.1. Introduction

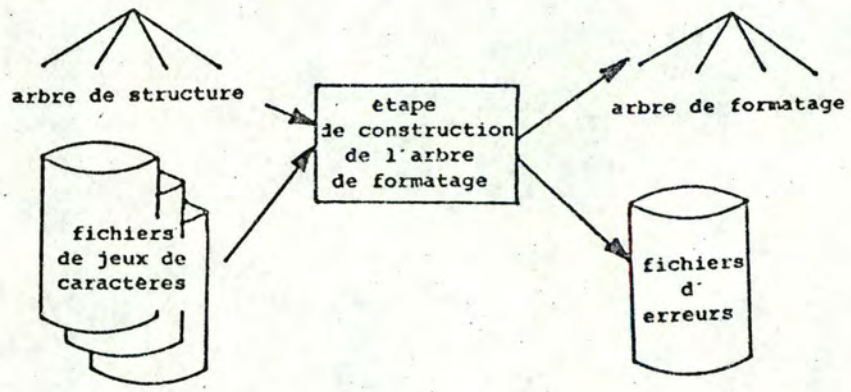
L'étape de construction de l'arbre de formatage aura comme but, à partir de l'arbre de structure, de construire un arbre qui sera une représentation de ce qui doit être imprimé. L'arbre de formatage sera une représentation PHYSIQUE du document.

L'étape de construction de l'arbre de formatage aura comme but, outre la mise en page en simple ou double colonne, la construction d'une table des matières, la réalisation de la pagination et la construction de l'index.

A ce stade, il sera nécessaire de disposer de fichiers contenant par jeu de caractères la taille de chacun des 128 caractères ASCII. Un jeu sera défini par une fonte et une police (pour plus de détails concernant les caractères, se référer au point 4.2.4.4).

Des erreurs pouvant être détectées lors de cette étape, le fichier d'erreurs, créé lors de l'étape de construction de l'arbre de structure, sera complété si nécessaire.

La figure 4.7 illustre cette étape.



Cette section comprendra :

- une description du contenu de l'arbre de formatage
- une figure représentant un arbre de formatage (figure 4.10), le texte source est le même que celui qui a servi pour la figure de représentation de l'arbre de structure.

4.3.4.2. Description du contenu de l'arbre de formatage

L'arbre de formatage comprendra un ensemble d'éléments correspondant à une disposition physique des éléments logiques de l'arbre de structure.
Les différents types d'éléments sont :

- le type document contenant N type page(s) (*)
- le type page contenant M type entité(s) (*)
- le type entité contenant P type ligne(s) (*)
- le type ligne contenant Q type bout de ligne(s) (*)
- le type bout de ligne contenant des caractères

(N, M, Q >= 1 et P >= 0 ; P = 0 dans le cas où l'on a une figure)

En premier lieu, les éléments de l'arbre de formatage devront comprendre des indications qui permettront leur positionnement.

Chaque élément de l'arbre se positionne par rapport à l'élément de niveau supérieur :

- les coordonnées de l'entité se détermine par rapport à la page
- les coordonnées des lignes se détermine par rapport à l'entité
- les coordonnées des bouts de lignes se détermine par rapport à la ligne.

C'est ainsi que dans la figure 4.8 (page suivante), figure illustrant le positionement des éléments les abcisses étant respectivement X pour l'abcisse vertical et Y pour l'abcisse horizontal, l'on aura le déplacement horizontal (y) et le déplacement vertical (x) par rapport à l'élément de niveau supérieur. Un élément sera en plus caractérisé par une largeur (delta_x) et une hauteur (delta_y). La figure 4.8a présente le positionnement d'une entité par rapport à une page ; la figure 4.8b présente le positionnement d'une ligne par rapport à une entité et la figure 4.8c présente le positionnement d'un bout de ligne par rapport à une ligne.

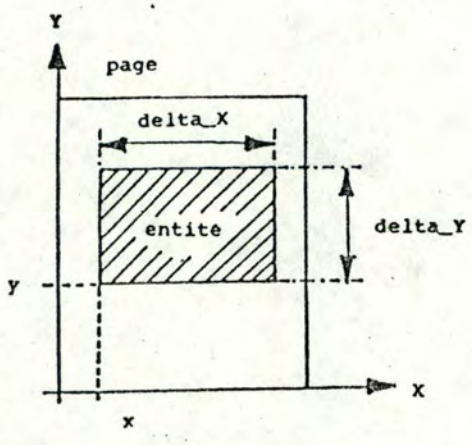


Fig. 4.8a.

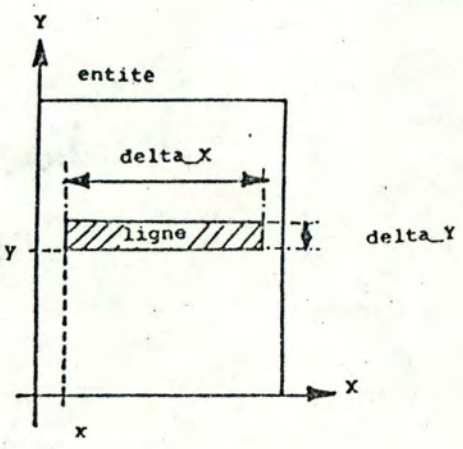


Fig. 4.8b.

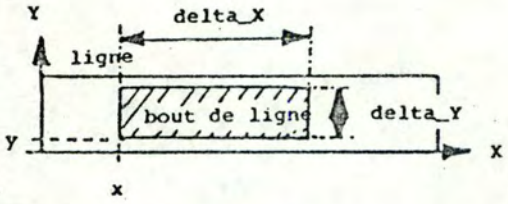


Fig. 4.8c.

En deuxième lieu, aux éléments de l'arbre, éléments de type bout de ligne, on devra associer un certain nombre d'indications typographiques pour l'impression, par exemple : le type de fonte, de soulignement, de police désiré ainsi que les caractères du texte.

En troisième lieu, à chacun des éléments de l'arbre de formatage sont associés des pointeurs afin de réaliser les liens entre ces éléments.

Chaque élément comprendra deux pointeurs :

- le premier pointeur contient l'adresse de l'élément suivant mais de type différent ; c'est ce que nous appelons "successeur"

- le deuxième pointeur contient l'adresse de l'élément suivant et de même type ; c'est ce que nous appelons "suivant"

La figure 4.9 (page suivante) illustre le contenu d'un élément de l'arbre de formatage.

TYPE D'ELEMENT	page,entite,ligne,bout de ligne
X	position verticale en nombre de points typographiques
Y	position horizontale en nombre de points typographiques
DELTA_X	largeur en nombre de points typographiques
DELTA_Y	hauteur en nombre de points typographiques
NOMBRE D'ELEMENT	integer
TYPE DE POLICE	10,11,14 ou 20
TYPE DE FONTE	gras,italique ou normal
TYPE DE SOULIGNEMENT	continu,disontinu,superieur, rien
STRING	caracteres du bout de ligne (maximum 128)
INTERLIGNE	nombre de points typographiques
POINTEUR SUCCESSEUR	pointeur vers l'element de même type
POINTEUR SUIVANT	pointeur vers l'element infereieure

Fig. 4.9. Description d'un élément de l'arbre

4.3.4.3. Exemple de représentation d'un arbre de formatage

La figure 4.10 illustre une représentation d'un texte sous la forme d'un arbre de formatage ; tout d'abord, la figure 4.10a présente le positionnement des éléments logique sur la page, ensuite la figure 4.10b présente une représentation de l'arbre de formatage. Le texte est celui de la figure 4.5. (page 4.10).

Remarque : les dimensions sont données en points typographiques ; dans l'exemple, les dimensions sont uniquement données à titre d'illustration.

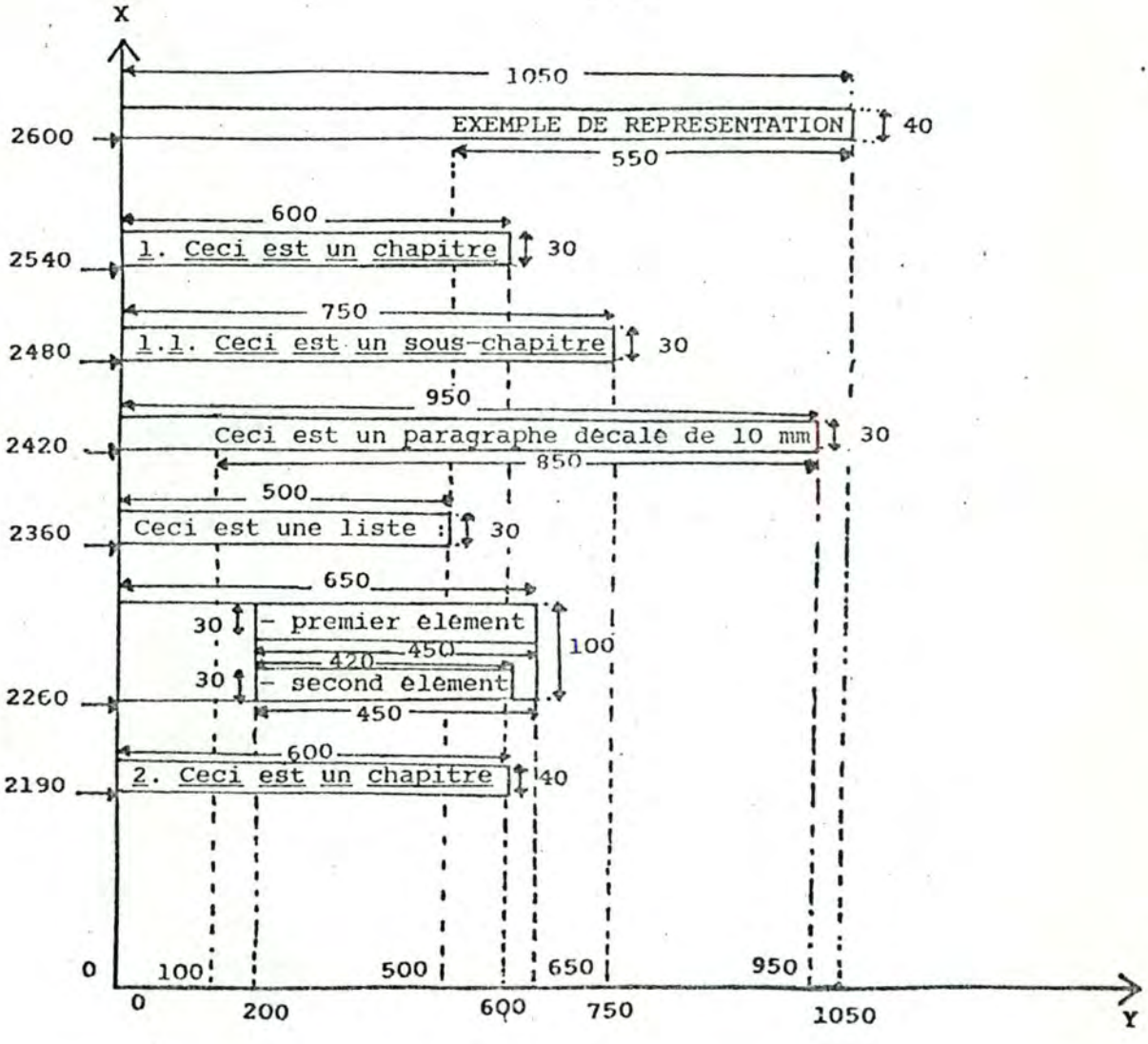
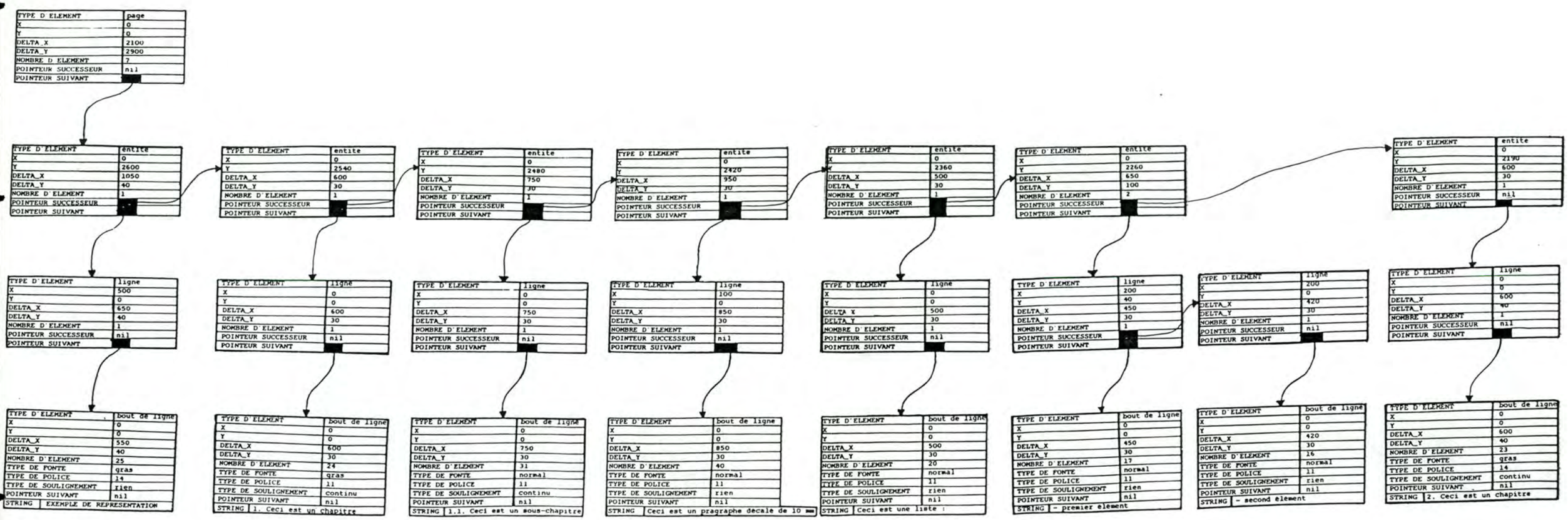


Fig. 4.10a.


TYPE D'ELEMENT | page
X | 0
Y | 0
DELTA_X | 2100
DELTA_Y | 2900
NOMBRE D'ELEMENT | 7
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2600
DELTA_X | 1050
DELTA_Y | 40
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2540
DELTA_X | 600
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2480
DELTA_X | 750
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2420
DELTA_X | 950
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2360
DELTA_X | 500
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2260
DELTA_X | 650
DELTA_Y | 100
NOMBRE D'ELEMENT | 2
POINTEUR SUCCESSEUR
POINTEUR SUIVANT
TYPE D'ELEMENT | entite
X | 0
Y | 2190
DELTA_X | 600
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 500
Y | 0
DELTA_X | 650
DELTA_Y | 40
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 0
Y | 0
DELTA_X | 600
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 0
Y | 0
DELTA_X | 750
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 100
Y | 0
DELTA_X | 850
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 0
Y | 0
DELTA_X | 500
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 0
Y | 0
DELTA_X | 200
DELTA_Y | 40
NOMBRE D'ELEMENT | 450
POINTEUR SUCCESSEUR | 30
POINTEUR SUIVANT | 1
TYPE D'ELEMENT | ligne
X | 200
Y | 0
DELTA_X | 420
DELTA_Y | 30
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | ligne
X | 0
Y | 0
DELTA_X | 600
DELTA_Y | 40
NOMBRE D'ELEMENT | 1
POINTEUR SUCCESSEUR | nil
POINTEUR SUIVANT
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 550
DELTA_Y | 40
NOMBRE D'ELEMENT | 25
TYPE DE FONTE | gras
TYPE DE POLICE | 14
TYPE DE SOULIGNEMENT | rien
POINTEUR SUIVANT | nil
STRING | EXEMPLE DE REPRESENTATION
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 600
DELTA_Y | 30
NOMBRE D'ELEMENT | 24
TYPE DE FONTE | gras
TYPE DE POLICE | 11
TYPE DE SOULIGNEMENT | continu
POINTEUR SUIVANT | nil
STRING | 1. Ceci est un chapitre
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 750
DELTA_Y | 30
NOMBRE D'ELEMENT | 31
TYPE DE FONTE | normal
TYPE DE POLICE | 11
TYPE DE SOULIGNEMENT | continu
POINTEUR SUIVANT | nil
STRING | 1.1. Ceci est un sous-chapitre
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 850
DELTA_Y | 30
NOMBRE D'ELEMENT | 40
TYPE DE FONTE | normal
TYPE DE POLICE | 11
TYPE DE SOULIGNEMENT | rien
POINTEUR SUIVANT | nil
STRING | Ceci est un pragraphe décalé de 10
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 500
DELTA_Y | 30
NOMBRE D'ELEMENT | 20
TYPE DE FONTE | normal
TYPE DE POLICE | 11
TYPE DE SOULIGNEMENT | rien
POINTEUR SUIVANT | nil
STRING | Ceci est une liste :
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 450
DELTA_Y | 30
NOMBRE D'ELEMENT | 17
TYPE DE FONTE | normal
TYPE DE POLICE | 11
TYPE DE SOULIGNEMENT | rien
POINTEUR SUIVANT | nil
STRING | - premier element
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 420
DELTA_Y | 30
NOMBRE D'ELEMENT | 16
TYPE DE FONTE | normal
TYPE DE POLICE | 11
TYPE DE SOULIGNEMENT | rien
POINTEUR SUIVANT | nil
STRING | - second element
TYPE D'ELEMENT | bout de ligne
X | 0
Y | 0
DELTA_X | 600
DELTA_Y | 40
NOMBRE D'ELEMENT | 23
TYPE DE FONTE | gras
TYPE DE POLICE | 14
TYPE DE SOULIGNEMENT | continu
POINTEUR SUIVANT | nil
STRING | 2. Ceci est un chapitre

4.3.5. Etape de création du fichier binaire dans le format FDD

L'étape de création du fichier binaire aura comme but, à partir de l'arbre de formatage, de créer un fichier binaire qui contiendra toutes les indications pour l'impression. La figure 4.11 illustre cette étape.

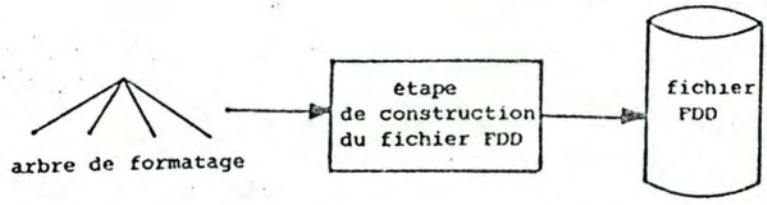


Fig. 4.11. Construction du format "FDD"

Le fichier binaire résultat est dans le format "FDD". La description de ce format se trouve dans l'annexe C. Il s'agit d'un document réalisé par R. HERSCH de l'E.P.F.L. qui décrit le format et la méthode pour le créer.

4.4. La structure des fichiers

4.4.1. Introduction

Les méthodes d'organisation que nous avons envisagées étaient soit le séquentiel, soit le séquentiel indexé. Les fichiers d'organisation séquentielle indexée peuvent contenir plusieurs types d'articles et une clé d'accès identifiante doit être définie. La gestion est assurée par des logiciels qui recouvrent des activités telles que la suppression, l'ajout, la modification ou la recherche d'articles. Le choix d'une organisation dépend du contenu du fichier et des besoins occasionnés par le traitement qui lui est associé.

Etant donné que le système s'exécute en mémoire centrale (pour permettre une grande rapidité d'exécution) et que le seul type d'accès que nous utilisons est la consultation, nous avons retenu l'organisation séquentielle.

Nous présenterons dans cette section les différents types de fichier :

- le fichier de données
- les fichiers des résultats
- les fichiers de tests
- les fichiers de jeux de caractères

4.4.2. Les fichiers de données

Le fichier de données ou fichier source contient le texte au kilomètre rentré au terminal par un utilisateur. Il se compose du texte proprement dit et des commandes de formatage dont la sémantique et la syntaxe sont décrites dans le manuel utilisateur ; c'est donc un fichier de caractères.

Chaque ligne dans le fichier est séparée de la suivante par un séparateur de ligne et le fichier est terminé par un caractère spécial d'END OF FILE. Le traitement opéré sur ce fichier est une lecture séquentielle pour exécuter une analyse syntaxique.

4.4.3. Les fichiers de résultats

Les fichiers de résultats sont au nombre de quatre :

- fichier en format "BINAIRE"

  Le fichier en format "BINAIRE" contient toutes les informations de l'arbre de formatage, d'informations nécessaires pour l'impression. Ces informations sont dans le format FDD (cfr. Annexe C).

- fichier en format "S" :

  Le fichier en format "S" sert pour le transfert entre le réseau EPNET et le réseau COBUS (cfr. chapitre 3 : chaîne d'impression). Il comprend les mêmes informations que le fichier binaire.

- fichier en format "SM" :

  Le fichier en format "SM" est directement lié à l'imprimante. C'est le format qui doit être chargé (cfr. chapitre 3 : chaîne d'impression).

- fichier d'erreurs :

  Le fichier d'erreurs comprend la liste des erreurs que l'utilisateur aura faites dans les commandes de formatage.

4.4.4. Les fichiers de tests

Les fichiers de tests sont construits après chaque étape du formatage. Ils s'obtiennent par des parcours récursifs des arbres de structure, de formatage, et du format "FDD". Leur contenu est détaillé à la section 4.7 : Méthode de test

4.4.5. Les fichiers de jeux de caractères

Chaque fichier comprend un entête qui se compose notamment du type de fonte, de la taille de la matrice du caractère, de l'adresse du premier et dernier caractère stocké. Pour chaque caractère sont spécifiées en binaire sa largeur, sa hauteur au-dessus et en-dessous de la ligne de base (*) et enfin une matrice que la figure 4.12 illustre.

Pour une police et une fonte déterminées, les caractères ont leurs dimensions fixées mais celles-ci varient suivant le type de caractères ; par exemple, un "M" aura une largeur plus grande qu'un "I".

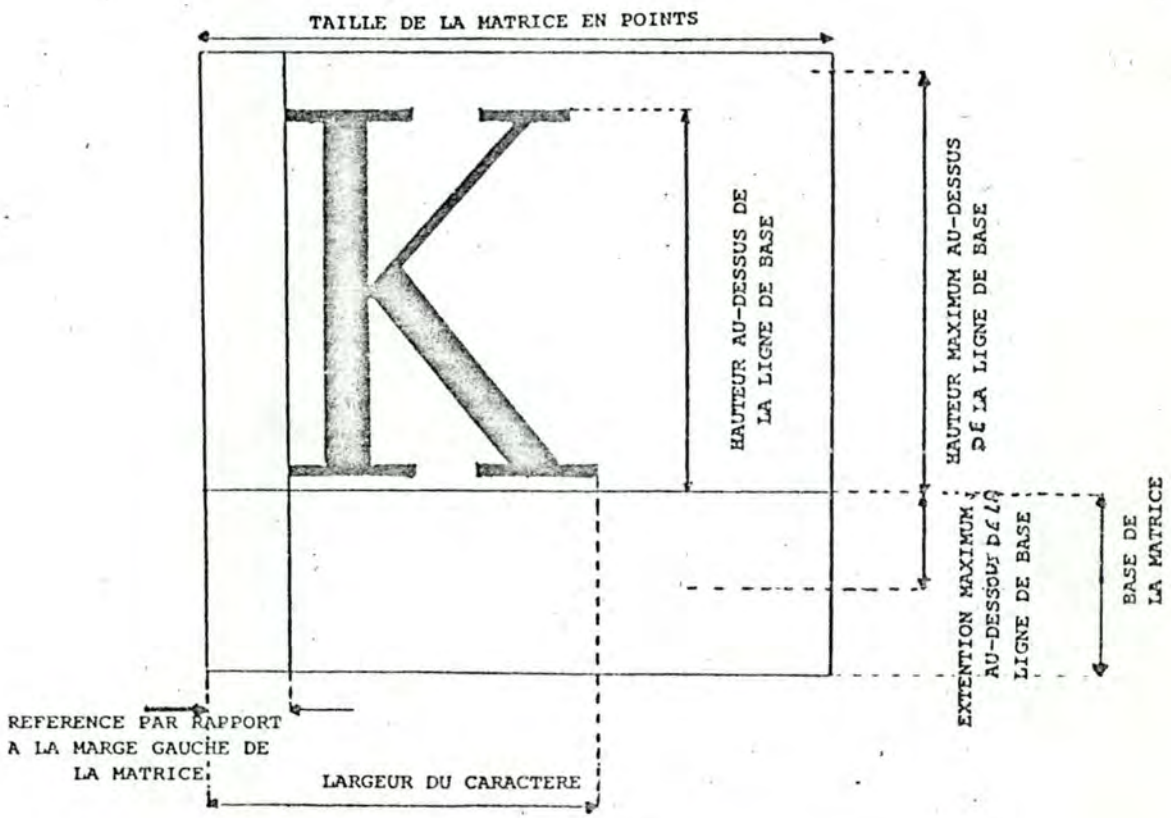


Fig. 4.12. Représentation matricielle d'un caractère

Les caractères sont donc représentés par leurs dimensions. Une autre approche consisterait à encoder le contour de chaque caractère. C'est ce qu'on appelle la représentation analytique. IL est possible de décrire le contour par des segments de droites ou des cercles. L'avantage de cette méthode est qu'un jeu de caractères d'un certain style est complètement décrit par la description analytique des contours de chaque caractère. Cette description est indépendante de la taille du caractère ; donc les caractères de n'importe quelle taille et d'un style donné peuvent être générés par leur représentation analytique.

Il y a donc deux possibilités de représenter les jeux de caractères. Soit par leur dimensions, soit par une description analytique. Nous avons choisi la première méthode car une étude réalisée par les assistants du laboratoire a démontré que la deuxième méthode n'était performante que pour les caractères dont la taille dépassait 16 points typographiques. A partir de cette limite, la taille mémoire pour stocker les caractères croit exponentiellement.

Etant donné que nous ne disposions que d'un nombre limité de jeux de caractères et que ceux-ci étaient tous de taille inférieure à 16 points typographiques, nous avons choisi la première représentation. Nous donnons dans le manuel utilisateur au point 6 les différentes tailles et styles de caractères disponibles sur la laser.

4.5. Architecture physique du système

4.5.1. Introduction

Si l'analyse fonctionnelle fait totalement abstraction de l'existence d'une configuration informatique quelconque, il n'en va pas de même de la définition d'une architecture.

Le projet sous étude ne considère pas l'implémentation d'un nouveau système mais concerne l'utilisation du matériel existant sur le lieu du stage. Le matériel impose des contraintes dans le cadre de l'identification des programmes et éventuellement dans la définition de la structure des fichiers.

Nous présentons dans cette section :

- un rappel sur les notions de fonctions, modules
- les raisons du choix de l'architecture
- la description de l'architecture

4.5.2. Rappel

Le principe de la décomposition modulaire se base sur la découpe fonctionnelle. Un certain nombre de fonctions "logiques" ont été définies lors de l'analyse fonctionnelle. Toutes les fonctions ne peuvent être considérées comme des modules séparés ; il faut redéfinir une architecture c'est-à-dire regrouper certaines fonctions ou en scinder d'autres. Cependant, la redéfinition d'une découpe doit suivre certains principes basés sur la correspondance entre fonctions et modules.

Les figures 4.13 et 4.14 illustrent ces principes [Van Lamsweerde 81]

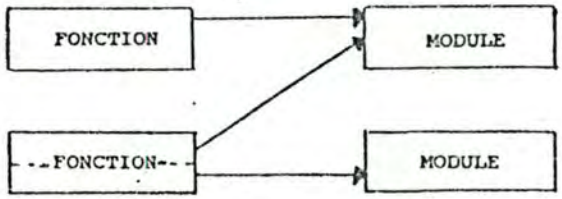


Fig. 4.13. Découpe fonction-module

Cependant une même fonction ne peut être intégrée dans deux modules différents dont un traite déjà une autre fonction.

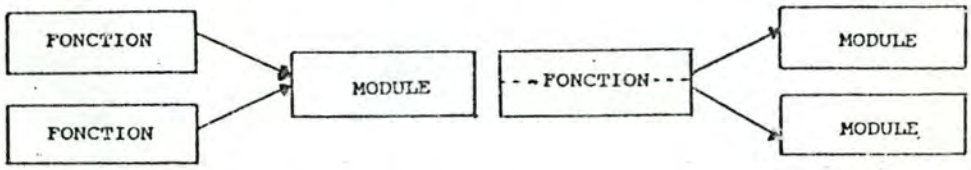


Fig. 4.14. Découpe fonction-module

L'architecture en modules répond également à des critères, parmi ceux-ci :

- la capacité de cacher de l'information (représentation des données et le mode de traitement sur ces données)
- le faible degré de couplage (minimiser les relations entre les modules, maximiser l'interdépendance)
- la forte cohésion interne (degré d'interdépendance logique entre conditions et actions au sein d'un même module)

4.5.3. Raisons du choix de l'architecture

Nous avons choisi l'architecture en fonction des considérations méthodologiques décrites dans l'introduction et en fonction de trois contraintes :

- respecter le principe de la décomposition fonctionnelle afin de garder une certaine cohérence du système. Etant donné que nous n'employons aucune fonction d'entrée/sortie ni de gestion de fichier, les modules décrits ne sont que le reflet des fonctions "logiques" de l'analyse fonctionnelle.

- localiser les modifications à apporter lors du changement de la configuration matérielle. Par exemple : choix d'un autre type d'imprimante (ces problèmes sont détaillés dans le chapitre 7)

- se donner une découpe telle que nous puissions implémenter graduellement le système

Le formateur étant implémenté en PASCAL, la notion de module est remplacée par la notion de procédure.
Pour pouvoir découper l'architecture en niveaux, nous avons utilisé les concepts suivants :

- programme coordinateur
- programme
- sous-programme
- sous-sous-programme
- fonction
- utilitaire

Ces concepts correspondent en fait à des procédures sauf le concept de programme coordinateur.

Chaque niveau de l'architecture devient plus complexe au fur et à mesure que l'on descend dans la hiérarchie des concepts cités ci-dessus.
C'est ainsi que :

- le premier niveau est constitué du programme coordinateur

- le deuxième niveau comprend les trois programmes appelés par le coordinateur. Ce niveau correspond aux étapes de création des 2 arbres et de l'étape de construction du format "FDD". Nous plaçons également à ce niveau le programme ayant comme but de créer le fichier binaire de résultat

- le troisième niveau comprend tous les sous-programmes c'est-à-dire les fonctions principales de chacune des étapes de la phase du formatage

- le quatrième niveau comprend les sous-sous programmes qui reprennent encore une découpe plus fine des étapes.

- le cinquième niveau est composé de toutes les fonctions élémentaires et spécifiques à un traitement

- le sixième niveau comprend tous les utilitaires. Par utilitaire, nous entendons des procédures PASCAL qui peuvent être utilisées par les niveaux supérieurs mais qui sont spécifiques à un programme déterminé

La figure 4.15 (page suivante) illustre la découpe ainsi que les procédures associées à chaque niveau.

NIVEAU 1

NIVEAU 2

NIVEAU 3

NIVEAU 4

NIVEAU 5

NIVEAU 6

PROGRAMME PRINCIPAL

ARBRESTRUCTURE
ARBREFORMATAGE
FDDTRAITEMENT

DEFELEMBASE
MODELEMBASE
RECLEX
REMPLIRMOT
TABLEAUCARACTERE
TRTABREFORMATAGE
MISEENPAGE
NUMEROTATION
TABLEMATIERE
CONSTRDIRECTORY
CONSTRPARTDIRECTORY
PARCOURS

TRAITMENTCLASSIQUE
UNECOLO
DEUXCOLO

PROMODBOOL
PROMODREDUIT
PROMODAUTRE
PROMARGE
PRODIVERS

CREERRECORD
LIEN
RECIMPL
RECHERCHERNIVEAU
RECHERCHERDIMENTION
PARCOURSFINNOTE
FINNOTEBASPAGE

NUMEROATIONFIG
NUMEROTATIONNOTE
NUMEROTAIONENTETE
IMPLEMLISTE

INIT
TRTMOT
MISEAJOUR
TRTCHANGEMENT
JUSTIFIE
TRTFINLIGNE
TRTCOMPLETAGE
TRTCENTRAGE

UPOSITIONY1
UPOSITIONY2
UPOSITIONY3
UCREERPAGE
BREAKPAGE
URESERVATIONNOTE
UPLACEMENTNOTE
UCASSURE

RESERVATIONNOTE
RESERVATIONFIGURE
PLACEMENTNOTE
PLACEMENTFIGURE
CREERPAGE
COLONNAGE
SAUTPAGE
SEULESURPAGE
POSITIONY1

GESTIONMATIERE
PROCENT
TRTTABLE

TRTPAGELIST
TRTPAGEDATA
TRTLIGNE

LECTURECARACTERE
LECDEUXPOINT
LECJUSTBLANC
LECSLAH
LECBLANC
DECODAGEVALEUR
EXPDIX
EXAMEN
CREERRECORD
GERRECORD
BINAIREDECIMALE
INCR
MISERECORD
TRANSFORMEWORD
REMPLIRWORD
MISEAMOINSUN
VALEUR

4.5.4. Description de l'architecture

Programme COORDINATEUR

Programme ARBRE DE STRUCTURE

Sous-programme DEFELEMBASE

Sous-programme MODELEMBASE

Fonction PROMODAUTRE
Fonction PROMODREDUITE
Fonction PROMODBOOL
Fonction PROMARGE
Fonction PRODIVERS

Sous-programme RECLEX

Fonction RECIMPL
Fonction RECHERCHERNIVEAU
Fonction RECHERCHERDIMENTION
Fonction CREERECORD
Fonction LIEN
Fonction FINNOTEBASPAGE
Fonction PARCOURSFINNOTE

Sous-programme REMPLIRMOT

Fonction NUMEROTATIONFIG
Fonction NUMEROTATIONNOTE
Fonction NUMEROTATIONENTETE
Fonction IMPLEMLISTE

Sous-programme PARCOURSARBSTRUC

Utilitaires :

LECTURECARACTERE, LECPOINT, LECDEUXPOINT, DECODAGEVALEUR, LECSLASCH, LECJUSTBLANC, EXPDIX, ERREUR

Programme ARBRE DE FORMATAGE

Sous-programme TABLEAUCARACTERE

Sous-programme TRTARBREFORMATAGE

Sous-sous-programme TRAITEMENTCLASSIQUE

Fonction INIT
Fonction TRTMOT
Fonction MISEAJOUR
Fonction TRTCHANGEMENT
Fonction JUSTIFIE
Fonction TRTFINLIGNE
Fonction TRTCOMPLETAGE

Sous-programe PARCOURSARBFORM

Sous-programme MISEENPAGE

Sous-sous-programme UNECOLO

Fonction UPOSITIONY1
Fonction UPOSITIONY2
Fonction UPOSITIONY3
Fonction UCREERPAGE
Fonction BREACKPAGE
Fonction URESERVATIONNOTE
Fonction UPLACEMENTNOTE
Fonction UCASSURE

Sous-sous-programme DEUXCOLO

Fonction RESERVATIONNOTE
Fonction RESERVATIONFIGURE
Fonction PLACEMENTNOTE
Fonction PLACEMENTFIGURE
Fonction CREERPAGE
Fonction COLONNAGE
Fonction CASSURE
Fonction SEULESURPAGE
Fonction SAUTPAGE

Sous-programme NUMEROTATION

Sous-progamme TABLEMATIERE

Fonction TRTTABLE
Fonction GESTIONMATIERE
Fonction PROCENT

Utilitaires :

GERERECORD, CRERRECORD, EXAMEN, BINAIREDECIMALE

Programme FDDTRAITEMENT

Sous-programme CONSTRDIRECTORY

Sous-programme CONSTRUCTDIRECTORY

Fonction TRTPAGELIST
Fonction TRTPAGEDATA
Fonction TRTLIGNE

Sous-programme PARCOURS

Utilitaires :

MISEAMOINSUN, MISERECORD, TRANSFORMWORD, REMPLIRWORD, VALEUR, INCR, DECODE

4.6. Spécification des modules

Les spécifications de tous les modules étant assez conséquentes, elles sont reprises dans l'annexe A. Les programmes PASCAL de chaque procédure sont repris dans l'annexe B.

Pour chaque module, nous définissons les éléments en entrée, les éléments en sortie, nous citons les pré-conditions et les post-conditions, les procédures appelantes et appelées, nous donnons un algorithme logique et nous définissons les variables internes

Les spécifications de l'annexe A s'adressent aux programmeurs, soit pour utiliser les propriétés d'un module et en développer d'autres, soit pour modifier ce module.

4.7. Méthodes de tests

Etant donné qu'il n'y a aucune partie interactive dans le système, nous avons établi la correspondance suivante : "un module = un programme". A cause de la philosophie des personnes avec lesquelles nous travaillions (écrire le plus rapidement possible des lignes de code et constater ensuite le résultat), les méthodes de tests ont été rigoureuses. Elles nous ont aidés à établir la présence d'erreurs et l'absence d'erreur.

Nous avons testé séparément chaque étape du formatage :

- l'étape de détermination des paramètres de présentation
- l'étape de construction de l'arbre de structure
- l'étape de construction de l'arbre de formatage
- l'étape de construction du fichier FDD

Pour réaliser cette opération, nous avons construit des fichiers de test pour chacune des étapes, ce qui nous a permis de comparer le contenu effectif des fichiers et le résultat que nous attendions et de détecter les différences.
C'est ainsi que :

- au niveau du test de l'étape de détermination des paramètres de présentation, une procédure avait comme but d'imprimer l'ensemble des paramètres et leur valeur.

- au niveau du test de l'étape de construction de l'arbre de structure, une procédure avait pour but de parcourir récursivement l'arbre et d'imprimer dans un fichier le contenu.

  La procédure s'appelle PARCOURSARBSTRUC, elle est spécifiée dans l'annexe A et se trouve dans l'annexe B, la figure 4.16 (page 4.34) illustre le procédé de test (on part d'un texte au kilomètre contenant des commandes de formatage, les commandes peuvent être correctes ou non, dans ce dernier cas elles sont soulignées et engendrent une "erreur" détectée ; le fichier de test proprement dit contient pour chaque élément logique : le type de l'élément, l'indication de centrage, ... ainsi que les mots associés , mots pour lesquels on associe les paramètres)

- au niveau du test de l'étape de construction de l'arbre de formatage, une précédure avait pour but de parcourir récursivement l'arbre et d'imprimer dans un fichier le contenu.

  La procédure s'appelle PARCOURSARBFORM, elle est spécifiée dans l'annexe A et se trouve dans l'annexe B, la figure 4.17 (page 4.35) montre un exemple du contenu du fichier de test (même principe que pour le test du contenu de l'arbre de structure)

- au niveau du test de l'étape de construction du fichier binaire, nous avions associé à chaque donnée un numéro. L'ajout d'une numérotation permet de tester facilement si la construction du format "FDD" s'est déroulée correctement.

Fig 4.16. Illustration des tests de l'étape de construction de l'arbre de structure

- Texte au kilomètre contenant un ensemble de commandes de formatage :

```
\\tab1 ; 3b
\do.
\ti.\\jus.\\pol : 1.TEST
\da.Lausanne, le 9 avril 83
\re.Tests des erreurs- exemple de
\\ct.texte au kilom\'etre\\cr.avec commandes de formatage, erreurs
\ch :niv: 1.\en.Debut
\par.Ceci est un\\jus.\\fon : tot.\\fon : . paragraphe permettant\\ligne.
\\soul : t.de tester\\dec : 45. \\dec : rer. \\mar : su : 45. \\marg : d : .
\\mar : inf : toto. \\poid : 45.l'arbre de structure .
\\too. \\list.tout d'abord\\impl : t.
\\do: 23.une\\de : f. liste\\\\de deux elements.
\pa.\\margigau : 79.\\margidroit : 12.ensuite un paragraphe
\pa.\\dec : 34.\\centr.centre\\cr. avec saut de ligne \\fon : ital.
 fonte italique\\pol :14. police 14\\fin:fon.retour fonte initiale
\\fin : pol. fin du paragraphe.
\pa.\\cen.Ensuite un paragraphe centre
\cha: niv : 3.\en. Ceci est un chapitre de niveau 3
\cha: niv : 2.\ent.chapitre niv 2.
\cha: niv: 3.\en. chapitre niv 3.
\cha : niv: 4.\en. chapitre niveau: 4.
```

- Fichier de test : contenu :

```
TYPE :       PAGE
X :        Ø Y :        Ø DELTAX :        Ø DELTAY :       ØLARGEURLIGNE :
      ØNOMBRELEMENT :                Ø
VALEUR BLANC :        Ø NOMBRE BLANC :        Ø

TYPE :      ENTITE
X :       94 Y :     2342 DELTAX :     1276 DELTAY :       2ØLARGEURLIGNE :
     1276NOMBRELEMENT :                1
VALEUR BLANC :        Ø NOMBRE BLANC :        Ø

TYPE :       LIGNE
X :       94 Y :        Ø DELTAX :       69 DELTAY :       2ØLARGEURLIGNE :
     1276NOMBRELEMENT :                1
VALEUR BLANC :       18 NOMBRE BLANC :        Ø

TYPE :   BOUTLIGNE
X :       94 Y :        Ø DELTAX :       69 DELTAY :       2ØLARGEURLIGNE :
     1276NOMBRELEMENT :                4
VALEUR BLANC :       18 NOMBRE BLANC :        Ø
TEST
TYPE :      ENTITE
X :        Ø Y :     2273 DELTAX :     1465 DELTAY :       22LARGEURLIGNE :
     1465NOMBRELEMENT :                1
VALEUR BLANC :        Ø NOMBRE BLANC :        Ø

TYPE :       LIGNE
X :      473 Y :        Ø DELTAX :      378 DELTAY :       22LARGEURLIGNE :
     1465NOMBRELEMENT :                1
VALEUR BLANC :       2Ø NOMBRE BLANC :        4

TYPE :   BOUTLIGNE
X :      473 Y :        Ø DELTAX :      378 DELTAY :       22LARGEURLIGNE :
     1465NOMBRELEMENT :               23
VALEUR BLANC :       2Ø NOMBRE BLANC :        4
Lausanne, le 9 avril 83
TYPE :      ENTITE
X :       94 Y :     2145 DELTAX :     1276 DELTAY :       31LARGEURLIGNE :
     1276NOMBRELEMENT :                2
VALEUR BLANC :        Ø NOMBRE BLANC :        Ø

TYPE :       LIGNE
X :       94 Y :       5Ø DELTAX :      911 DELTAY :       31LARGEURLIGNE :

...
```

Fig. 4.17. Illustration des tests de l'étape de construction de l'arbre de formatage

- Texte au kilomètre contenant un ensemble de commandes de formatage :

```
\do.
\ti.\\jus.\\pol : 1.TEST
\da.Lausanne, le 9 avril 83
\re.Tests des erreurs- exemple de
\\ct.texte au kilom\'etre\\cr.avec commandes de formatage, erreurs
\ch :niv: 1.\en.Debut
\par.Ceci est un\\jus.\\fon : tot.\\fon : . paragraphe permettant\\ligne.
\\soul : t.de tester\\dec : 45. \\dec : ref. \\mar : su : 45. \\marg : d : .
\\mar : inf : toto. \\pold : 45.l'arbre de structure .
\\fo9. \\list.tout d'abord\\impl : t.
\\de: 23.une\\de : f. liste\\\de deux elements.
\pa.\\marg:gau : 79.\\marg:droit : 12.ensuite un paragraphe
\pa.\\dec : 34.\\centr.centre\\cr, avec saut de ligne \\fon : ital.
 fonte italique\\pol :14. police 14\\fin:fon.retour fonte initiale
\\fin : pol. fin du paragraphe.
\pa.\\cen.Ensuite un paragraphe centre
\\cha: niv : 3. Ceci est un chapitre de niveau 3
```

- Liste des erreurs rencontrées :

```
RENCONTRE DE: "\\C_" OU: "_" N'EST PAS DEFINI CORRECTEMENT
\ct
ERREUR DANS LA SPECIFICATION DE LA FONTE
\fo
ERREUR DANS LA SPECIFICATION DE LA FONTE
\fo
ERREUR DANS LA SPECIFICATION DU DECALAGE D'UN PARAGRAPHE
TYPE DE MARGE NON RECONNU
\ma
ERREUR DANS LA SPECIFICATION DE LA MARGE DROITE
TYPE DE MARGE NON RECONNU
\ma
ERREUR DANS UNE COMMANDE D'IMPLEMENTATION
\to
ERREUR DANS LA SPECIFICATION DU DECALAGE D'UNE LISTE
RENCONTRE DE: "\\C_" OU: "_" N'EST PAS DEFINI CORRECTEMENT
\ch
```

- Fichier de test : contenu :

```
-------------------------------------------
    DOCUMENT  cet element n est PAS CENTRE

 INDICATEUR DE SAUT DE PAGE  : false
 MARGE GAUCHE :          0
 MARGE DROITE :          0
-------------------------------------------
       TITRE  cet element n est PAS CENTRE

 INDICATEUR DE SAUT DE PAGE  : false
 MARGE GAUCHE :         10
 MARGE DROITE :         25
TEST##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      GRAS
TYPE DE POLICE DU MOT :          14
INDICATEUR DE SAUT DE LIGNE : false
-------------------------------------------
       DATE  cet element est centre

 INDICATEUR DE SAUT DE PAGE  : false
 MARGE GAUCHE :         10
 MARGE DROITE :         25
Lausanne, ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :  ITALIQUE
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
le ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :  ITALIQUE
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
9 ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :  ITALIQUE
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
avril ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :  ITALIQUE
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
83##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :  ITALIQUE
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
```

```
-------------------------------------------
    ABSTRACT  cet element n est PAS CENTRE

 INDICATEUR DE SAUT DE PAGE  : false
 MARGE GAUCHE :         10
 MARGE DROITE :         25
Tests ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
des ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
erreurs- ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
exemple ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
detexte ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
au ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
kilometreavec ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : false
commandes ##
TYPE DE SOULIGNEMENT DU MOT :        RIEN
TYPE DE FONTE DU MOT :      NORMAL
TYPE DE POLICE DU MOT :          11
INDICATEUR DE SAUT DE LIGNE : vrai
```

...

4.8. Description des donnees.

4.8.1. Introduction.

Nous avons distingue deux classes de donnees, les donnees de stucture decrites dans une notation Bacchus et les donnees d'implementaton decrites en francais.

Nous prenons comme convention de representer le "ou" par le symbole "|" dans la representation Bacchus. De plus les differents elements seront decrits suivant une methode descendante (de l'element complexe vers l'element atomique).

4.8.2. Données de structure.

4.8.2.1. Introduction.

Les données de structure sont les composants utilisés dans la construction des arbres. Deux types de données de structure sont utilisés, les données logiques - données manipulées par l'arbre de structure - représentant la structure logique d'un document (paragraphe, entête,..) et les données physiques - données manipulées par l'arbre de formatage - représentant la structure physique d'un document (page, ligne,..)

4.8.2.2. Données logiques.

‹document logique› ::= ‹éléments d'identification›
‹corps de document› ‹table des matières› ‹index›
‹bibliographie›

‹elements d'identification› ::=
‹element d'identification› |
‹element d'identification› ‹elements d'identification›

‹element d'identification› ::= ‹titre› | ‹auteur› | ‹resume› | ‹date›

‹titre› ::= ‹mots›

‹auteur› ::= ‹mots›

‹resume› ::= ‹mots›

‹date› ::= ‹mots›

‹corps de document› ::= ‹chapitre› |
‹chapitre› ‹corps de document›

‹chapitre› ::= ‹entete› ‹corps de document› |
‹entete› ‹éléments de corps›

‹éléments de corps› ::= ‹élément de corps› |
‹élément de corps› ‹éléments de corps›

‹élément de corps› ::= ‹paragraphe› | ‹liste› |
‹figure› | ‹note bas de page›

‹entete› ::= ‹mots›

‹paragraphe› ::= ‹mots›

‹liste› ::= ‹separateur› ‹élément d'une liste› |
‹separateur› ‹élément d'une liste› ‹ liste›

‹élément d'une liste› ::= ‹mots›

‹figure› ::= ‹dessin› ‹texte de figure›

‹dessin› ::= élément externe

‹texte de figure› ::= ‹mots›

‹note bas de page› ::= ‹mots›

‹table des matières› ::= ‹vide› |
‹entête de table› ‹corps de table›

‹entête de table› ::= ‹mots›

‹corps de table› ::= ‹mots›

‹index› ::= ‹vide› | ‹entête d'index› ‹corps d'index›

‹entête d'index› ::= ‹mots›

‹corps d'index› ::= ‹mots›

‹bibliographie› ::= ‹vide› |
‹entête de bibliographie› ‹corps de bibliographie›

‹entête de bibliographie› ::= ‹mots›

‹corps de bibliographie› ::= ‹mots›

‹mots› ::= ‹mot› | ‹mot› ‹mots›

‹mot› ::= ‹suite de séparateurs› ‹suite de caractères ASCII› |
‹suite de caractères ASCII› ‹suite de séparateurs›

‹suite de séparateurs› ::= ‹vide› | ‹separateur› |
‹separateur› ‹suite de séparateurs›

‹suite de caractères ASCII› ::= ‹vide› |
‹caractères ASCII› |
‹caractères ASCII› ‹suide de caractères ASCII›

‹séparateur› ::= ";" | "," | "!" | "´" | "`" | "?" |
"+" | "-" | "*" | "_" | "/" | "
"|"

‹caractère ASCII› ::= caractère dont on a retiré
l'ensemble des séparateurs.

‹vide› ::= 0

4.8.2.3. Données physiques.

‹document physique› ::= ‹page› | ‹page› ‹document physique›

‹page› ::= ‹entité› | ‹entité› ‹page›

‹entité› ::= ‹ligne› | ‹ligne› ‹entité› |
‹espace reservé›

‹espace reservé› ::= zone déterminée par une
hauteur et une largeur qui est destinée à
insérer une figure.

‹ligne› ::= ‹bout de ligne › | ‹bout de ligne› ‹ligne›

‹bout de ligne› ::= ‹mots›

Le bout de ligne correspond à l'élément atomique, tous les mots ont les mêmes caractéristiques de formatage.

4.8.3. Données d'implémentation typographique .

- BIBLIOGRAPHIE : répertoire des écrits références dans le document
- CENTRAGE : texte cadré par rapport à la marge gauche et à la marge droite
- COLONNAGE : texte réalisé en une ou deux colonnes par page
- DECALAGE DE PARAGRAPHE : nombre de millimètres de décalage de la première ligne d'un paragraphe
- FONTE : style de caractères : italique, normal, gras
- IMPLEMENTATION D'UNE LISTE : type d'élément désiré comme premier caractère d'un élément d'une liste : point, tiret, étoile, ..
- IMPRESSION REDUITE : indication que l'on ne désire qu'un nombre limité de feuille(s) imprimée(s)
- INDEX : table alphabétique de noms cités dans le document accompagnés de références
- INTERENTITE : nombre de millimètres a laisser entre deux entités
- INTERLIGNE : nombre de millimètres a laisser entre deux lignes
- JUSTIFIE : lignes traitées de façon à avoir un alignement à gauche et à droite
- LARGEUR DE FEUILLE : nombre de millimètres de la feuille en largeur
- LONGUEUR DE FEUILLE : nombre de millimètres de la feuille en longueur
- MARGE DROITE : nombre de millimètres laissés à la droite du texte
- MARGE INFERIEURE : nombre de millimètres laissés vides en bas de page
- MARGE GAUCHE : nombre de millimètres laissés vides à la gauche du texte
- MARGE SUPERIEURE : nombre de millimètres laissés vides en haut de la page
- PAGINATION : mise d'un numéro sur chacune des pages du document

- POLICE : grandeur des caractères

- SAUT DE PAGE : indication de passage à la page suivante

- SAUT DE LIGNE : indication de passage à la ligne suivante

- SOULIGNEMENT : méthode de mise en évidence, différents types de soulignement : continu, discontinu, supérieur

- TABLE DES MATIERES : énumération des chapitres, des questions traitées dans le document

- TYPE DE FEUILLE : genre de feuille pour laquelle on désire un formatage, le type de feuille implique une hauteur et une largeur de feuille

Chapitre 5 :

MANUEL UTILISATEUR

Chapitre 5:

MANUEL UTILISATEUR

Le Manuel utilisateur du formateur FTTL a été formaté à l'aide du formateur que nous avons réalisé et imprimé sur l'imprimante laser CANON LBP-10 à l'E.P.F.L.

Par manque de temps, nous n'avons pu réaliser ce manuel entièrement à Lausanne (le chapitre 1 étant réalisé sur le formateur NROFF à l'INSTITUT).

Chapitre 6 :

ETUDE COMPARATIVE ENTRE LES FORMATEURS "JUSTIF", "NROFF" ET "FTTL"

Chapitre 6:

ETUDE COMPARATIVE ENTRE LES FORMATEURS "JUSTIF", "NROFF" ET "FTTL"

6.1. Introduction

Dans ce chapitre seront presentées les commandes des formateurs JUSTIF (utilisé à l'E.P.F.L.), le formateur NROFF (utilisé à l'Institut d'Informatique) et FTTL. Les formateurs seront présentés dans cet ordre.
Les commandes sont regroupées par rubriques ; les différentes rubriques proposées sont :

- les modes
- le format de la page
- la mise en page
- le texte
- l'alphabet utilisé
- la numérotation
- les sauts
- les définitions
- le soulignement

Pour le formateur JUSTIF [SANDERS 81], les commandes doivent être insérées dans le texte source en minuscule et entièrement.

Pour le formateur NROFF [Ossanna 74], chaque commande doit se trouver en début de ligne. Une distinction sera faite entre les commandes définies dans le jeu de macros -MVARIO [Simpson 77] (les commandes de -MVARIO sont en majuscules) et les commandes de base (les commandes de base sont en minuscules). Un certain nombre de commandes ne seront pas citées en raison de leur utilité surtout technique.

Pour le formateur FTTL, les commandes peuvent être en minuscule ou en majuscule et les deux premiers caractères de la commande suffisent à l'identifier. La synthaxe bacchus (présentée dans le Manuel Utilisateur) sera utilisée pour décrir les commandes et une distinction sera faite entre le niveau global et le niveau local. Un format "standard" est défini par défaut pour chacun des éléments logiques.

Remarque : les commandes ont été regroupées en neuf rubriques de la façon la plus précise possible, mais vu l'optique différente de raisonnement des formateurs présentés, les commandes peuvent ne pas être "placées" dans la rubrique la plus judicieuse ; de plus, une même commande pourra se retrouver dans plusieurs rubriques.

6.2. Etude comparative des commandes

6.2.1. Les modes

JUSTIF

\justifie. texte justifié à gauche et à droite
\drapeaud. ou \drapd.
texte justifié à droite
\drapeaug. ou \drapg.
texte justifié à droite
\centre. texte centré

NROFF

COMMANDES DE BASE :

.fi	remplissage des lignes
.nf	inhibition de .fi
.ad	ajustement des lignes sur la largeur maximale
.na	inhibition de la commande .da
.ce N	centrage des N lignes suivantes.

COMMANDES DE -MVARIO

.DS C	ligne centrée
.DS L	ligne alignée à gauche
.DS I	ligne indentée
.DS R	ligne doublement indentée

FTTL

Niveau global :

\\JUSTIFIE : <ELEMENT LOGIQUE>.
\\CENTRE : <ELEMENT LOGIQUE>.
tous les éléments du genre <ELEMENT LOGIQUE> seront soit centrés, soit justifiés

Niveau local :

\\JUSTIFIE.
\\CENTRE.
l'élément <ELEMENT LOGIQUE> courant aura la forme justifiée ou centrée indépendamment de ce qui est défini au niveau global

6.2.2. Le format de la page

JUSTIF

\marge.	modifie la largeur de la marge gauche, produisant un décalage de tout le texte vers la droite
\gauche.	modifie la largeur de la marge gauche et la largeur du texte de facon à ce que la marge droite soit conservée
\large.	modifie la largeur d'impression du texte
\newlarge.	définit trois valeurs de largeur
\newmarge.	définit trois valeurs de marge
\margsup.	modifie la largeur de la marge supérieure
\marginf.	modifie la largeur de la marge inférieure
\longpage.	modifie la longueur de la page

NROFF

COMMANDES DE BASE :

.pl N	définition de la longueur de la page
.po +/-N	chaque ligne sera decalée de N caractères vers la droite
.ne N	définition de la marge supérieure
.mk a	le nombre de lignes sur la page est stocké dans un registre a
.rt -N	si N est positif : place N lignes à partir du sommet de la page si N est négatif : place N lignes après la ligne courante

COMMANDES DE -MVARIO :

.TP string	indentation d'un titre
.RS	indentation augmentée de 8 positions
.RE	indentation diminuée de 8 positions
.SE et .SQ	indente de 8 positions (marqueur de début et de fin de zone)
.RP	diminution d'une indentation

FTTL

Niveau global :

\\MARGE : SUPERIEURE : <VALEUR>.
\\MARGE : INFERIEURE : <VALEUR>.

la marge supérieure et/ou inférieure du document aura toujours la valeur redéfinie au niveau local indépendamment de ce qui est défini au niveau global.

\\MARGE : GAUCHE : <ELEMENT LOGIQUE> : <VALEUR>.
\\MARGE : DROITE : <ELEMENT LOGIQUE> : <VALEUR>.

pour tous les éléments du genre <ELEMENT LOGIQUE> sera défini une nouvelle marge droite et/ou gauche.

\\TYPE FEUILLE : <TYPEFEUILLE>.

redéfinition d'un nouveau type de feuille à formater, à chaque type correspond une largeur et une longueur de feuille.

Niveau local :

\\MARGE : GAUCHE : <VALEUR>.
\\MARGE : DROITE : <VALEUR>.

l'élément <ELEMENT LOGIQUE> courant aura une valeur pour la marge gauche et/ou la marge droite différente de celle qui est définie au niveau global.

6.2.3. La mise en page

JUSTIF

\newtab.	pour définir des colonnes de tabulation
\newind.	
\newtab.	pour définir des colonnes d'indentation
\tab n.	pour définir des colonnes de tabulation décimale
\ind n.	pour accéder à la nième colonne de tabulation
\ind n.	pour accéder à la nième colonne d'indentation
\ind xx mm.	pour indenter de xx mm
\dtab.	pour accéder à la colonne suivante de tabulation décimale
\interligne.	pour changer l'interligne
\lignes.	pour fixer le nombre de lignes par page

NROFF

COMMANDES DE BASE :

.ls N	initialisation de l'interligne à N lignes
.sp N	saut de N lignes
.ns	activation du "space mode", inhibition des commandes .sp et .bp
.rs	restauration du .sp et du .bp
.il N	mise à N caractères de la longueur de la ligne
.in N	mise à N caractères de l'indentation
.ti n	définition d'une indentation temporaire pour N ligne(s)
.ta N	définition d'une tabulation initialisée à la valeur N
.tc c	réinitialisation d'une tabulation
.lc c	réinitialisation d'une tabulation
.fc ab	intialiation de délimiteurs de champs

COMMANDES DE -MVARIO :

rien

FTTL

Niveau global :

\\INTERLIGNE : ‹ELEMENT LOGIQUE› : ‹VALEUR›.
l'interligne de l'élément ‹ELEMENT LOGIQUE› courant aura la valeur redéfinie.

\\ENTITE : ‹VALEUR›.
redéfinition de la valeur moyenne du nombre de mm à définir entre deux entités, la valeur effective est comprise entre 76 % de cette valeur et 126 % (raison : "l'élasticité entre entités")

Niveau local :

\\INTERLIGNE : ‹VALEUR›.
au sein de l'élément courant ‹ELEMENT LOGIQUE›, l'interligne aura la valeur définie indépendamment de ce qui est défini au niveau global.

6.2.4. Le texte

JUSTIF

\pousse. ou \ecarte. ou \écarte.
: pour aligner le restant de la ligne sur la marge droite

\insere.
: pour obtenir une ligne par la répétition d'une chaîne d'au plus 7 caractères

\indice.
: pour obtenir une chaîne imprimée plus bas que la ligne courante

\exp.
: pour obtenir une chaîne imprimée plus haut que la ligne courante

\up.
: pour un demi interligne vers le haut

\down.
: pour un demi interligne vers le bas

\back.
: équivalent à un back space

\exact.
: transmet le reste de la ligne telle qu'elle

\com.
: pour une ligne de commentaires

\print.
: pour forcer l'impression de ce qui précède

\titre.
: pour renforcer la ligne qui suit

\ctitre.
: comme \titre mais avec centrage

\trait.
: pour obtenir un trait sur toute la largeur de la page

NROFF

COMMANDES DE BASE :

.ex
: fin de texte

.it N
: définit la longueur du titre

.tl 'left''right''center'
: le titre doit être centré, aligné à gauche ou centré

.ig
: les lignes sont ignorées jusqu'à la rencontre de .GI

COMMANDES DE -MVARIO :

.TR
: titre en haut de page

.TL string
: ligne de titre

.CE
: fin de chapitre

.CP N string
: numérotation avec numéro de chapitre en haut de page

.PI N
: insertion d'une figure, réservation de n lignes

.DS TT string
: titre en haut de page

.FS
: debut de note bas de page

.FE
: fin de note bas de page

.LS
: début de zones à imprimer telles quelles

.LE
: fin de zones à imprimer telles quelles

.P1
: nouveau paragraphe

.SF
: début de figure

.EF
: fin de figure

.SH string
: section non numérotée

.DS
: début de display

.DE
: fin de display

FTTL

Niveau global :

\\DECALAGE : LISTE : <VALEUR>.
redéfinition du décalage d'une liste (décalage par rapport à la marge gauche)

\\DECALAGE : PARAGRAPHE : <VALEUR>.
redéfinition du décalage de la première ligne d'un paragraphe

\\FONTE : <ELEMENT LOGIQUE> : <TYPE DE FONTE>.
\\SOULIGNEMENT : <ELEMENT LOGIQUE> : <TYPE DE SOULIGNEMENT>.
\\POLICE : <ELEMENT LOGIQUE> : <TYPE DE POLICE>.
\\MARGE : GAUCHE : <ELEMENT LOGIQUE> : <VALEUR>.
\\MARGE : DROITE : <ELEMENT LOGIQUE> : <VALEUR>.
\\CENTRE : <ELEMENT LOGIQUE> .
\\JUSTIFIE : <ELEMENT LOGIQUE>.
redéfinition des parmètres de présentation pour l'élément <ELEMENT LOGIQUE> spécifié.

Niveau local :

\EXPOSANT. ou \UP.
ce qui suit se trouvera en exposant et ce jusqu'à ce que l'on rencontre \down.
\DOWN. ou \BAS.
ce qui suit se trouvera en indice jusqu'à ce que l'on rencontre \up.
\\DECALAGE : X.
modification au sein d'une liste ou d'un paragraphe du décalage

On défini toujours une élément logique sous la forme suivante :

\<ELEMENT LOGIQUE>.

cette définition a deux conséquences :

1. créer un nouvel élément logique du type défini
2. terminer l'élément logique précédent.

Exception à cette règle : la note bas de page se termine

- soit par la commande définissant l'élément logique suivant
- soit par la structure : \FIN NOTE BAS DE PAGE.

Comme cela a déjà été mentionné, il est possible de redéfinir des paramètres de présentation pour chaque élément logique et cela indépendamment de ce qui est défini au niveau global.

6.2.5. L'alphabet utilisé

JUSTIF

\hel. ou \normal.
caractres à chasse variable
\messenger. caractères IBM courrier 12
\italique. caractères à chasse variable et italique
\grec. caractères grecs

NROFF

Rien de signalé dans les commandes de NROFF et de -MVARIO, d'où la nécessité d'utiliser des commandes de la SANDERS.

FTTL

Les caractères sont en chasse variable. Trois types de caractères pourront être utilisés : normal, italique ou gras. Trois types de police pourront être utilisés (cfr. documents annexes).

Niveau global :

\\FONTE : ‹ELEMENT LOGIQUE› : ‹TYPE DE FONTE›.
\\POLICE : ‹ELEMENT LOGIQUE› : ‹TYPE DE POLICE›.
redéfinition de la fonte et/ou de la police pour l'élément ‹ELEMENT LOGIQUE› spécifié.

Niveau local :

\\FONTE : ‹TYPE DE FONTE›.
à partir de cette instruction, les caractères suivants auront la fonte définie par ‹TYPE DE FONTE› indépendamment de ce qui est défini au niveau global.

\\FIN : FONTE.
cette instruction a comme but de restaurer la fonte définie au niveau global pour l'élément logique courant. (Cette instruction fait suite à l'instruction \\FONTE:‹TYPE DE FONTE›.)

\\POLICE : ‹TYPE DE POLICE›.
à partir de cette instruction, les caractères suivants auront la police définie par ‹TYPE DE POLICE› indépendamment de ce qui est défini au niveau global.

\\FIN : POLICE.
cette instruction a comme but de restaurer la police définie au niveau global pour l'élément logique courant. (Cette instruction fait suite à l'instruction \\POLICE : ‹TYPE DE POLICE›.)

6.2.6. La numérotation

JUSTIF

\date.	pour définir la date en toute lettre
\cdate.	pour définir la date en toute lettre en anglais
\chap.	
\num.	
\mnum.	
\gnum.	
\par.	
\snum.	
\msnum.	
\gsnum.	
\ssnum.	
\page.	pour passer à la page suivante
\fig.	
\sfig.	
\stable.	
\liste.	pour une liste simple avec des tirets au début des paragraphes. Indentation de 10 mm
\lista.	pour une liste alphabétique et indentation au début des articles
\listnum.	pour une liste numérotée
\finliste.	pour stopper une liste

NROFF

COMMANDES DE BASE :

.pn N	la page suivant la courante aura la numérotation N

Il est possible d'utiliser et de définir des registres pour numéroter automatiquement des sections, paragraphes, lignes, ...

.nr a +/- N - M	définition ou modification du registre a ; il est initialisé à la valeur +/- N et l'argument M est utilisé pour l'auto-incrémentation
.nc c	un nombre de caractères est intialisé à c
.ar	chiffre arabe
.ro	chiffre roman minuscule
.RO	chiffre roman majuscule
.nm +/-N M/S/I	permet la numérotation des lignes, la première est numérotée à N
.np M/S/I	initialisation ou réinitialisation des paramètres de numérotage

COMMANDES DE -MVARIO :

.NE	réinitialisation du compteur des notes
.NO	numéro de note
.NP	numéro de paragraphe

.PN N	réinitialisation du numéro de page
.PP N	réinitialisition du numéro des paragraphes
.NH N	numéro d'une section

FTTL

Le numérotage est effectué automatiquement pour :

- les entêtes de chapitre de niveau 1 à 4, ou lors de la rédaction du texte source l'utilisateur spécifie le numéro du niveau de l'entête (qui est celle du chapitre en considération)

  exemple :

  structure :

  \CHAPITRE : NIVEAU : 3.\ENTETE. Ceci est ...

  résultat :

  3.2.4. Ceci est ...

- les figures avec une référence au sein du texte

  exemple :

  structure :

  ...\FIG 34,42.Ceci est une figure\FIN : FIGURE....

  résultat :

  * dans le texte, on aura :

  ...(FIG. 3)...

  * en dessous de la place réservée pour la figure, on aura :

  Fig. 3. Ceci est une figure

- les notes bas de page avec une référence au sein du texte

  exemple :

  structure :

  ...\NOTE.Ceci est une note\FIN : NOTE....

  résultat :

  * dans le texte, on aura :

  ...(nb.3)...

  * au bas de la page, on aura :

  NB. 1. Ceci est une note

- la liste , chaque élément étant séparé par un triple boa. Au niveau de l'implémentation sont disponibles le point, le tiret et l'étoile, une amélioration consisterait a permettre les listes numérotées numériquement et alphanumériquement.

Niveau global :

\\PAGINATION : SUPERIEUR.
\\PAGINATION : INFERIEUR.
définit que le document va être paginé, le numéro de page sera soit en haut de page (SUPERIEUR) soit en bas de page (INFERIEUR)

\\NUMERO : NOTE : <VALEUR>.
initialisation du compteur de notes

\\NUMERO : PAGE : <VALEUR>.
initialisation du compteur de pages

\\NUMERO : ENTETE : NIVEAU : 1 : <VALEUR>.
initialisation du compteur des entêtes de niveau 1

\\NUMERO : FIGURE : <VALEUR>.
initialisation du compteur des figures

Niveau local :

rien

6.2.7. Les sauts

JUSTIF

\cr.	provoque un saut de ligne
\ff.	provoque un simple saut de page
\BS.	back space
\BL.	remonte d'une ligne

NROFF

COMMANDES DE BASE :

.br	saut de ligne
.bp	saut de page

COMMANDES DE -MVARIO :

.BP	passe à la page
.SO	passe à la ligne
.S1	passe à la ligne et saute une ligne
.S N	passe à la ligne et saute N lignes

FTTL

\\PAGE. ou \\FF.
l'élément logique courant est forcé à la page suivante du document

\\LIGNE. ou \\CR.
les caractères suivants de l'élément logique courant seront forcés à la ligne suivante

deux commandes de saut de ligne inscrites l'une derrière l'autre et séparée d'un ou de plusieurs caractères blancs provoquent un saut de deux lignes (de même, trois commandes séparées chacune d'un ou plusieurs caractères blancs provoquent un saut de trois lignes et ainsi de suite)

Exemple :

\\CR. \\CR. \\CR.

6.2.8. Les définitions

JUSTIF

\rename. ou \define.
pour redéfinir une macro.

\savdef. pour sauver un fichier les macros définies dans le texte et celles chargées par \getdef.

\substitute. pour obtenir la substitution automatique d'un caractère par un autre

NROFF

COMMANDES DE BASES :

.de xx	définit ou redéfinit une macro de nom "xx". Le corps de la macro commence à la ligne suivante et se termine par une ligne commencant par ".."
.am xx	ajout de la macro de nom "xx"
.ds xx string	définit une string. La string "string" est définie comme une macro ayant un ou deux caractère(s)
.as xx string	ajout d'une string "string" de nom "xx"
.rm xx	réinitialise une macro ou une string
.wh -n xx	après la nième ligne rentrée, insertion de la macro de nom "xx"
.ch -n -m et .ch xx -m	change l'emplacement de la macro de la ligne n à la ligne m
.em xx	fin de spécification d'une macro
.di xx	re définition de la macro xx à partir de l'endroit courant
.da xx	inhibition de la commande .di xx

COMMANDES DE -MVARIO :

rien

FTTL

L'usage des macros est permis. Les macros doivent être décrites dans un fichier indépendant du fichier contenant le texte au kilomètre. Elles ont comme but de subtituer une chaîne de caractères par une autre.

Format de définition d'une macro :

"chaîne à remplacer" = "chaîne remplaçante"

Le caractère délimiteur est le caractère ", une chaîne à remplacer a au maximum 10 caractères et une chaîne remplaçante, au maximum 500 caractères. Aucune exception n'est faite quant au contenu des chaînes en dehors de cette limitation de longueur.

6.2.9. Le soulignement

JUSTIF

__ le texte se trouvant entre "__" est souligné

NROFF

COMMANDES DE BASE :

.ul n soulignement des n lignes suivantes

COMMANDES DE -MVARIO :

.UL n soulignement des n lignes suivantes

FTTL

Niveau global

\\SOULIGNEMENT : <ELEMENT LOGIQUE> : <TYPE DE SOULIGNEMENT>.
redéfinition du type de soulignement pour l'élément <ELEMENT LOGIQUE> courant

Niveau local

\\SOULIGNEMENT : <TYPE DE SOULIGNEMENT>.
\\FIN : SOULIGNEMENT.
les caractères situés entre ces deux instructions seront soulignés suivant le type défini et cela indépendamment de ce qui a été défini au niveau global. L'instruction de fin de soulignement restaure le soulignement défini au niveau global

6.2.10. Commandes diverses

Listes de commandes n'entrant pas dans les rubriques ci-dessus.

NROFF

COMMANDES DE BASE :

a. La conversion de caractères :

.ec c	le "basic control character" est initialisé au point ou au caractère "c"
.ee c	le caractère "escape" devient le "boa" ou le caractère "c"
.c2 c	le caractère "not breakcharacter" est initialisé à "c" ou réinitialisé à "'"
.tr abcd	remplacement de caractères dans l'ordre de lecture

b. La césure

.nh	pas de césure
.hy	césure
.hc c	caractère d'indication de césure

c. Les instructions conditionnelles d'acceptation de lignes

.if c anything
.if !c anything
.if N anything
.if !N anything

si la condition est vraie ou si le registre N est positif, alors la ligne est acceptée entièrement sinon le reste de la ligne est ignorée

d. Les différents environnements possibles

.ev N	définition de N environnements

e. Les fichiers annexes

.so filename	connection à un autre fichier avec retour
.nx filename	connection à un autre fichier sans retour

FTTL

Niveau global :

\\TABLE DES MATIERES : NOMBRE DE NIVEAU
\\TABLE DES MATIERES.
demande pour avoir une table des matières qui contient les entêtes de chapitre. Le nombre de niveaux peut être spécifié (au cas où il ne le serait pas, le nombre de niveaux serait : 2)

\\FONTE : TABLE DES MATIERES : <TYPE DE FONTE>.
\\POLICE : TABLE DES MATIERES : <TYPE DE POLICE>.
\\INTERLIGNE : TABLE DES MATIERES : <VALEUR>
\\SOULIGNEMENT : TABLE DES MATIERES : <TYPE DE SOULIGNEMENT>
\\MARGE : GAUCHE : TABLE DES MATIERES : <VALEUR>
\\MARGE : DROITE : TABLE DES MATIERES : <VALEUR>
\\CENTRE : TABLE DES MATIERES.
\\JUSTIFIE : TABLE DES MATIERES.
spécification des paramètres de présentation de la table des matières.

\\COLONNAGE.
demande pour avoir un document formaté en deux colonnes

\\INDEX.
demande pour avoir un index.

\\IMPRESSION REDUITE : <VAL1> , <VAL2>.
demande pour que n'imprimer que les feuilles dont le numéro est compris entre VAL1 et VAL2.

\\UNE PAGE.
demande pour avoir les éléments d'identification (titre du document, nom des auteurs, date, résumé) centrés sur la première page (page non numérotée même au cas où la numérotation est demandée)

6.3. Conclusions

Le formateur réalisé possède une grande partie des commandes des deux autres. NROFF étant très puissant quant à la création de macros (instructions conditionnelles).

Principaux avantages du formateur réalisé :

a. Au niveau des commandes :

- toutes les données numériques de l'utilisateur se font en millimètres
- libre choix de la majuscule ou de la minuscule laissé à l'utilisateur pour les commandes
- possiblité de commandes abrégées (permettant d'accepter des erreurs de frappes dans les commandes)
- commandes en français (aucun symbole)
- commandes insérées "n'importe-où" dans le texte
- non nécessité d'aller à la ligne pour une commande
- facilement adaptable dans une langue étrangère
- gestion des erreurs du langage de commande (indication d'un numéro d'erreur)

b. Au niveau de la structure :

- structure d'un document comme base (définition de type d'élément logique)
- numérotation automatique des chapitres (en fonction du niveau), des figures et des notes bas de page
- assignation de paramètres de présentation par type d'élément logique
- notion de "colle" entre éléments logiques (section 3.4.4.2, page 3.13)

c. Au niveau du grand choix de possibilités :

- grande possibilité d'ajouter de nouveaux jeux de polices et de fontes (y compris des fontes de caractères grecs)
- possibilité de n'imprimer qu'un nombre réduit de feuille(s) - indication du numéro de la première page et de la dernière page à imprimer
- plusieurs types d'implémentation de liste disponible
- possibilité de formater pour différents types de feuille
- possibilité d'avoir les éléments d'identification sur une page
- plusieurs types de fonte, police, soulignement disponibles
- possibilité d'insérer du graphisme
- plusieurs sortes de pagination disponibles
- possiblité d'avoir un texte en double colonne
- création automatique d'une table des matières (tous les paramètres de présentation des éléments de la table ainsi que le nombre de niveaux d'entêtes peuvent être donnés par l'utilisateur)

Améliorations et modifications à apporter (des indications quant à l'implémentation de ces différents éléments seront données dans la suite du manuel) :

- notion de tableau et de tabulation
- définition de compteurs auxiliaires avec possiblités de réinitialisation
- possibilité de paginer, de numéroter les notes et les figures par chapitre
- inscription de la date automatique
- possibilité d'avoir une référence sur chaque feuille
- possibilité d'insérer un texte COMMENTAIRE dans le texte source
- possibilité d'avoir un index

Chapitre 7 :

EVALUATION - PORTABILITE

Chapitre 7:

EVALUATION - PORTABILITE

7.1. Introduction

Ce chapitre a comme but de présenter notre conclusion personnelle sur le travail réalisé et d'indiquer les améliorations que l'on pourrait y apporter.

Il se compose de trois parties :

- dans une première partie, nous donnons une évaluation du travail réalisé ainsi qu'une liste d'améliorations à apporter au formateur afin d'accroître ses possibilités ; pour chacune des ces améliorations, nous donnons une façon de les implémenter

- dans une deuxième partie, nous explicitons les modifications à apporter au formateur afin de pouvoir utiliser n'importe quel type d'imprimante

- dans une troisième partie, nous citons les problèmes que nous avons rencontrés lors de la réalisation de ce travail

7.2. Evaluation et amélioration

7.2.1. Introduction

Cette section se compose de deux parties :

- la première partie reprend l'évaluation du formateur réalisé. Pour cette évaluation, nous citons une liste des principaux avantages du formateur

- la seconde partie reprend les améliorations à apporter au formateur. Pour chaque amélioration, nous décrivons le format de la commande associé ainsi que la façon de l'implémenter. Une remarque reprendra deux améliorations concernant la phase d'édition et d'impression.

7.2.2. Evaluation

Nous avons réalisé un formateur qui rassemble un grand nombre des avantages des formateurs existants.

Nous citons entre autres :

- structure d'un document comme base
- toutes les données numériques de l'utilisateur se font en millimètres
- assignation de paramètres de présentation par type d'élément logique
- libre choix de la majuscule ou de la minuscule laissé à l'utilisateur pour les commandes
- possiblité de commandes abrégées (permettant d'accepter des erreurs de frappes dans les commandes)
- commandes en français (aucun symbole)
- commandes insérées "n'importe-où" dans le texte
- non nécessité d'aller à la ligne pour une commande
- facilement adaptable dans une langue étrangère
- grande possibilité d'ajouter de nouveaux jeux de polices et de fontes (y compris des fontes de caractères grecs)
- numérotation automatique des chapitres (en fonction du niveau), des figures et des notes bas de page
- possibilité de n'imprimer qu'un nombre réduit de feuille(s)
- possibilité d'avoir plusieurs types d'implémentation de la liste
- possiblité d'avoir les éléments d'identification centrés sur une page (page non numérotée)
- plusieurs types de fonte, de police, de soulignement, d'implémentation de liste disponibles
- possiblité d'insérer du graphisme (en faisant une référence à un graphique se trouvant dans un autre fichier)
- plusieurs types de pagination disponibles (en haut ou en bas de page)
- possiblité d'avoir un texte en double colonne

- création automatique d'une table des matières (les paramètres de présentation des éléments de la table ainsi que le nombre de niveaux d'entêtes peuvent être mentionnés par l'utilisateur)

- possibilité pour l'utilisateur de définir ses propres commandes en utilisant les macros

- gestion des erreurs (indication d'un numéro d'erreur) des commandes

7.2.3. Amélioration

7.2.3.1. Introduction

Par manque de temps, nous n'avons pu implémenter toutes les fonctions que nous aurions désiré réaliser.

Les différentes fonctions qu'il serait intéressant d'implémenter sont les suivantes :

- la notion de tableau
- la notion de tabulation
- la notion de commentaire
- la notion de date automatique
- la notion de référence
- la notion d'index
- la notion de compteurs auxiliaires
- la notion de numérotation par chapitre

Pour chacune de ces notions, nous donnons dans la description, le format de la commande et dans l'implémentation, la façon de l'implémenter.

7.2.3.2. Notion de tableau

- Description

  Se définir une commande au niveau global de type :

  \\TABLEAU : VALEUR1 : VALEUR2.

  Cette commande aurait comme but de créer un tableau avec VALEUR1 représentant le nombre de colonnes désiré et VALEUR2 représentant la largeur en millimètre d'une colonne.

- Implémentation

  Il faudrait créer un élément logique de type tableau ; il comprendrait les mêmes paramètres de présentation et les mêmes pointeurs que les autres éléments logiques ; il comprendrait cependant deux paramètres supplémentaires :

- un paramètre qui serait un ARRAY[1..10] OF INTEGER et qui représenterait le nombre de colonnes désiré par l'utilisateur

- un paramètre qui serait un entier et qui représenterait la largeur d'une colonne

La construction du tableau et le remplissage de ses colonnes pourraient s'implémenter d'une manière similaire à l'implémentation d'une liste. Chaque élément d'une colonne serait séparé par (par exemple) trois boas et l'utilisateur passerait d'une colonne à la suivante en introduisant 4 boas.

7.2.3.3. Notion de tabulation

- Description

Se définir une commande au niveau global de type :

\\TABULATION.

Cette commande aurait comme but d'amener les caractères suivant la commande à la tabulation définie suivante

- Implémentation

La façon d'implémenter cette commande serait similaire à la façon d'implémenter la commande du décalage de la première ligne d'un paragraphe.

7.2.3.4. Notion de commentaire

- Description

Se définir deux commandes au niveau local de type :

\COMMENTAIRE.

et

\FIN : COMMENTAIRE.

Les caractères situés entre ces deux commandes seraient ignorés lors du formatage

- Implémentation

Lors de l'étape de construction de l'arbre de structure, on se bornera à lire les caractères du fichier contenant le texte au kilomètre jusqu'à ce que la fin du commentaire ou la fin du texte soit détectée, et ce sans effectuer de traitement dessus.

7.2.3.5. Notion de date automatique

- Description

Se définir une commande au niveau local de type :

\DATE : AUTOMATIQUE.

Cette commande aurait comme but d'imprimer la date du système.

- Implémentation

Cette commande s'implémenterait de la même manière que la commande de création de la date sauf que l'utilisateur ne devrait pas rentrer la date car celle-ci serait remplacée par la date du système (nb. plusieurs formats d'impression pourraient être disponibles : année-mois-jour ou jour-mois-année ou ...) .

7.2.3.6. Notion de référence

- Description

Se définir deux commandes au niveau global de type :

\\REFERENCE : SUPERIEURE : "NOM".

et

\\REFERENCE : INFERIEURE : "NOM".

Ces commandes auraient comme but d'indiquer en haut de page ou en bas de page une chaîne de caractères "NOM".

- Implémentation

Créer une procédure qui aurait comme but de parcourir

l'arbre de formatage ; à chaque rencontre d'une page, elle créerait une entité qui comprendrait la chaîne de caractères spécifiée par l'utilisateur ; cette entité serait reliée à la page et ses coordonnées déterminées en fonction du fait du positionnement de la référence : en haut ou en bas de page.

7.2.3.7. Notion d'index

- Description

Se définir une commande au niveau local de type :

\INDEX.

Cette commande aurait comme but d'annoter chaque mot suivi de cette commande ; l'ensemble des mots annotés serait listé suivant leur occurrence dans le texte au kilomètre et à chaque mot serait associé son numéro de page ; la liste serait reprise en fin de document.

- Implémentation

Il faudrait ajouter un champ au record du mot, par exemple : "index" qui serait un booléen. Lors de la construction de l'arbre de formatage, il y aurait construction d'un bout de ligne qui comprendrait le mot destiné à être repris dans l'index (même principe que pour la gestion de changement de fonte ou de police ou de soulignement). Il faudrait ensuite créer une procédure qui serait appelée après la procédure de NUMEROTATION et qui aurait comme but de parcourir l'arbre de formatage ; pour chaque bout de ligne annoté rencontré, elle créerait une entité et gérerait ses coordonnées sur la page (même principe que pour la création d'une table des matières).

7.2.3.8. Notion de compteurs auxiliaires

- Description

Se définir une commande au niveau global de type :

\COMPTEUR : "NOM" : VALEUR : INCREMENT.

Cette commande aurait comme but de définir un compteur de nom "NOM", d'initialiser sa valeur à VALEUR et de déterminer la valeur de l'incrémentation à INCREMENT.

Se définir une commande au niveau local de type :

\COMPTEUR : "NOM".

Cette commande aurait comme but de demander l'impression du compteur et d'incrémenter celui-ci de la valeur de l'INCREMENT.

- Implémentation

Il faudrait définir une matrice 10 * 3 qui reprendrait pour chacun des 10 compteurs son nom, sa valeur initiale et la valeur de son incrémentation. Lors de la construction de l'arbre de structure, la rencontre d'une commande globale de définition d'un compteur aurait comme but d'initialiser ses valeurs ; la rencontre d'une commande de niveau local de définition d'un compteur aurait comme but d'ajouter la valeur du compteur au dernier élément logique créé et d'incrémenter le compteur de la valeur de son incrément.

7.2.3.9. Notion de numérotation par chapitre

- Description

Se définir une commande au niveau global de type :

\\NUMEROTATION : CHAPITRE.

Cette commande aurait comme but de numéroter les pages, les figures et les notes par chapitre. Par exemple la troisième figure du cinquième chapitre aurait le numéro : 5.3.

- Implémentation

Au niveau des procédures NUMEROTATION, NUMEROTATIONFIG, NUMEROTATIONNOTE, il faudrait introduire le niveau du chapitre courant et initialiser les variables servant à la numérotation en fonction du numéro du chapitre courant.

7.2.4. Remarques

Pour améliorer l'utilisation du formateur, il serait intéressant d'apporter une amélioration au niveau de la phase d'édition et au niveau de la phase d'impression.

- au niveau de la phase d'édition, on pourrait établir une correspondance entre les touches de fonctions et les commandes du formateur. Par exemple, la touche de fonction F1 sur un terminal VISUAL 200 correspondrait à une demande de soulignement.

- au niveau de la phase d'impression, il faudrait créer un interface qui gérerait tous les transferts et tous les chargements des différents fichiers afin de permettre l'envoi de la commande d'impression d'un document sur l'imprimante laser directement à partir du VAX.

7.3. Portabilité

Suivant l'imprimante utilisée, un interface doit être réalisé. Cet interface a pour but de transformer le fichier binaire dans le format FDD en un fichier comprenant les commandes de l'imprimante.

Dans le cas spécifique de l'imprimante SANDERS, une justification étant assurée par l'imprimante, il faudrait, en fonction de la grandeur minimale du caractère utilisé par l'imprimante, déterminer la valeur que celui-ci doit prendre, comme valeur minimale (dans la procédure de justification, la grandeur du caractère blanc correspondait à une certaine proportion de la grandeur du caractère "i" - la grandeur est déterminée par le jeu de caractères associés : fonte et police, la constante PROPORTIONBLANC déterminait la proportion).

Il faut nécessairement disposer pour une imprimante donnée de fichiers contenant pour chaque jeu de caractères (comme nous l'avons déjà dit, un jeu détermine une fonte et une police) de la taille des caractères ; chacun des fichiers doit avoir un format binaire (les bytes de 256 à 384 contenant la largeur des caractères ;les bytes de 385 à 512 contenant la hauteur au-dessus de la "base-line" ; les bytes de 513 à 640, la hauteur en dessous de la "base-line").

7.4. Problèmes rencontrés

Dans cette section, nous citons les problèmes que nous avons rencontrés pour la réalisation du mémoire, sur le lieu de notre stage et aux FACULTES.

A. Pour la réalisation du mémoire :

- difficulté de réunir de l'information sur le sujet traité : nous avons écrit à plusieurs constructeurs en Belgique, seuls IBM et PHILIPPS ont donné une réponse intéressante

B. Problèmes rencontrés à l'E.P.F.L.

- charge du VAX : le temps d'exécution, de compilation et même d'édition était très long ; de plus la taille mémoire dont nous disposions était nettement insuffisante ; ce dernier problème a été résolu au mois de décembre par l'ajout de deux disques supplémentaires

- problème au niveau du langage : lorsque nous sommes retournés à Pâques pour améliorer et compléter les programmes, la version du compilateur PASCAL avait été changée

- problème de logiciel : certains programmes extérieurs au problème de formatage proprement dit n'existaient pas ; c'est le cas notamment des programmes de décodage que nous avons du implémenter

- difficultés dans les opérations de transfert VAX - COBUS : il était fréquent que le fichier transmis était "amputé" de quelques lignes ou que chaque ligne du fichier transmis perdait ses deux ou trois premiers caractères

- difficulté lors du chargement du fichier en format "SM" et des fichiers de jeux de caractères : l'opération de chargement et d'envoi à l'impression sur l'imprimante laser se constituait d'une séquence d'une dizaine d'instructions ; il fallait souvent recommencer cette séquence avant d'obtenir une impression correcte

- taille mémoire insuffisante : l'interface, gérant les pages destinées à être imprimées, ne peut actuellement stocker plus de dix pages ; lors de la rédaction du manuel utilisateur, nous avons du le scinder en plusieurs fichiers. Chaque fichier, résultant du formatage, ne devait contenir plus de dix pages

- disponibilité constante des assistants : les assistants étaient constamment disponibles pour résoudre les différents problèmes cités ci-dessus

- disponibilité du matériel : le matériel sur lequel nous travaillions était constamment disponible, ce qui nous a permis le travail de nuit (notamment pour éviter les problèmes de charge)

C. Aux Facultés :

- problème lié à l'utilisation de l'imprimante SANDERS (problème de RUBAN).

CONCLUSION

CONCLUSION

Le but de ce travail était de concevoir et de réaliser un formateur de document.

La méthode de travail que nous avons suivie se rapproche beaucoup de celle reçue à l'institut d'informatique.

Dans une première étape, nous avons effectué une étude de l'évolution du problème de formatage en présentant les principaux formateurs caractéristiques.

C'est en fonction de cette étude et des contraintes matérielles que nous avons rencontrées sur le lieu de notre stage et des besoins de l'E.P.F.L que nous nous sommes fixé les caractéristiques du formateur que nous avons développé.

Dans une deuxième étape, nous avons expliqué la démarche d'analyse du système à réaliser. Cette étape est constituée de l'analyse fonctionnelle et de l'analyse organique.

Dans une troisième étape, nous avons rédigé un manuel utilisateur à l'aide du formateur que nous avons réalisé.

Dans une quatrième étape, nous avons fait une étude comparative entre le système réalisé et deux autres formateurs que nous avons dû couramment employer. Il s'agit des formateurs "JUSTIF" (utilisé à l'E.P.F.L) et "NROFF" utilisé à l'institut pour rédiger ce mémoire.

Dans une dernière étape, nous avons évalué le travail proposé en présentant les problèmes (de conception et de matériel) rencontrés sur le lieu de notre stage, et en soumettant une liste d'améliorations. Pour chacune de ces améliorations, nous proposons une idée pour l'implémenter (entre autres, la possibilité d'utiliser une telle conception de formatage pour un autre type d'imprimante par exemple une imprimante SANDERS).

DICTIONNAIRE DE DONNEES

DICTIONNAIRES DE DONNEES

l'arbre de formatage : nom donné à la représentation physique d'un document ; un arbre de formatage se compose d'élements de type page, de type entité, de type ligne, de type bout de ligne

l'arbre de structure : nom donné à la représentation logique d'un document ; un arbre de structure se compose d'éléments logiques

l'auteur : type d'élément logique correspondant au nom de la (ou des) personnes qui a (ou ont) fait un document

batch : mode d'exécution ne requérant aucune intervention de l'utilisateur

le boa : caractère ASCII : la barre oblique arrière ("\")

le centrage : paramètre de présentation d'un élément logique qui spécifie que le texte est cadré par rapport à la marge gauche et à la marge droite

le chapitre : type d'élément logique correspondant à une division d'un livre

le colonnage : paramètre de présentation d'un document qui spécifie que le texte doit être réalisé en une ou deux colonnes par page

la date : type d'élément logique correspondant à l'indication du jour, du mois et de l'année

le décalage : paramètre de présentation d'un élément logique de type paragraphe ou liste qui spécifie le nombre de millimètres d'indentation de la première ligne d'un paragraphe ou l'indentation de tous les éléments d'une liste

un document : moyen de communication entre deux personnes pouvant être un livre, une revue, une lettre, ... ; nous avons associé la notion de document à celle de livre

l'élément logique : élément constitutif d'un document

l'entête : type d'élément logique correspondant à un titre de chapitre

la figure : type d'élément logique correspondant à un dessin servant à la représentation

la fonte : paramètre de présentation d'un élément logique qui spécifie le type de style de caractères (italique, gras ou normal)

la hauteur d'une figure : paramètre de présentation d'un élément logique de type figure qui spécifie le nombre de millimètres de la figure en hauteur

l'implémentation d'une liste : paramètre de présentation d'un élément logique de type liste qui spécifie que chaque élément de la liste commence par un point, un tiret ou une étoile

une impression réduite : paramètre de présentation d'un document qui spécifie la(es) page(s) qui devra(ont) être imprimée(s)

l' index : paramètre de présentation d'un document qui spécifie que le document devra comporter une liste de termes avec une référence

l'interentité : paramètre de présentation d'un document qui spécifie le nombre de millimètres à laisser entre deux éléments logiques

l'interligne : paramètre de présentation d'un élément logique qui spécifie le nombre de millimètres à laisser entre deux lignes d'un même élément logique

la largeur d'une figure : paramètre de présentation d'un élément logique de type figure qui spécifie le nombre de millimètres en largeur d'une figure

la ligne de base : axe de coordonnées permettant de calculer les dimensions d'un caractère

la liste : type d'élément logique correspondant à une suite de noms

la marge droite : paramètre de présentation d'un élément logique qui spécifie le nombre de millimètres laissés à la droite du texte

la marge gauche : paramètre de présentation d'un élément logique qui spécifie le nombre de millimètres laissés à la gauche du texte

la marge inférieure : paramètre de présentation d'un document qui spécifie le nombre de millimètres laissés vides en bas de page

la marge supérieure : paramètre de présentation d'un document qui spécifie le nombre de millimètres laissés vides en haut de page

le mot : élément atomique de l'arbre de structure qui se constitue de caractères du texte au kilomètre et de paramètres de présentation pour ces caractères

la note bas de page : type d'élément logique correspondant à une remarque en bas de feuille

une page (...) : paramètre de présentation d'un document qui spécifie que les éléments d'identification du document se trouveront centrés sur une page non numérotée

la pagination : paramètre de présentation d'un document qui spécifie qu'un numméro sera mis sur chacune des pages du document

le paragraphe : type d'élément logique correspondant à une section d'un chapitre

les paramètres de présentation : paramètres associés à un élément logique ou à un document, spécifiant à la fois les caractéristiques typographiques des caractères et les caractéristiques concernant le contenu du document (table des matières, ...)

la police : paramètre de présentation d'un élément logique qui spécifie la grandeur des caractères (en point typographique)

le résumé : type d'élément logique correspondant à un abrégé, à un sommaire

le soulignement : paramètre de présentation d'un élément logique qui spécifie la méthode de mise en évidence, différents types de soulignement : continu, discontinu, supérieur

la table des matières : paramètre de présentation d'un document qui spécifie que le document comprendra une ou plusieurs page(s) comprenant l'énumération des chapitres, des questions traitées dans le document avec pour chacun de ces éléments une référence

le texte au kilomètre : suite de caractères sans retour de chariot

le titre : type d'élément logique correspondant au nom d'un livre

le type bout de ligne : élément atomique de l'arbre de formatage, qui comprend des caractères du texte source (caractères ayant les mêmes caractéristiques typographiques et se trouvant sur une même ligne), des paramètres de présentation et le placement des coordonnées par rapport à l'élément de type ligne le comprenant

le type de feuille : paramètre de présentation d'un document qui spécifie le genre de feuille pour laquelle on désire un formatage, le type de feuille implique une hauteur et une largeur de feuille

le type entité : élément de l'arbre de formatage correspondant à un élément logique de l'arbre de structure, qui comprend les coordonnées de placement par rapport à la page et une référence vers les éléments de type ligne

le type ligne : élément de l'arbre de formatage correspondant à une ligne (unité physique) d'un élément de type entité, qui comprend les coordonnées de placement par rapport à l'entité et une référence vers les éléments de type bout de ligne

le type "livre" : ouvrage en prose ou en vers, de quelque étendue

le type page : élement de l'arbre de formatage correspondant à une page (unité physique), qui comprend une référence vers les éléments de type entité

typographique : type d'impression désirée pour un caractère, par exemple le caractère sera dans la fonte X, dans la police Y, sera souligné . . .

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[Allen, Nix et Perlis 81]

Allen T., Nix R., Perlis A.
"A Hierarchical Document Editor"
Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices 16(6) : 74 - 81
Spring/Summer 1981

[Beatty, Chin et Moll 79]

Beatty J.C., Chin J.S., Moll H.F.
"An Interactive Documentation System"
SIGGRAPH '79 Proceedings, Computer Graphics 13(2) : 71 - 82
August 1979

[Berns 68]

Berns G.M.
"The FORMAT Program"
IEEE Transactions on Engineering Writing and Speech EWS - 11(2) : 85 - 91
August 1968

[Berns 69]

Berns G.M.
"Description of FORMAT"
Communications of the ACM 12(3) : 141 - 146
March 1969

[Berns et Ehrman 71]

Berns G.M., Ehrman J.R.
"FORMAT, A Text Processing Program"
SLAC Report 135
Standford Linear Accelerator Center, July 1971

[BYTE MAGAZINE 81]

BYTE MAGAZINE
"Special Issue on Smalltalk"
Byte Magazine 6(8)
August 1981

[Chamberlain 81]

Chamberlain D.C., King J.C., Slutz D.R., Todd S.J.P., Wade B.W.
"An Interactive System for Document Composition"
Proceedings of the ACM SIGPLAN SIGOA A Symposium on TEXT Manipulation, SIGPLAN Notices 16(6) : 82 - 91
June 1981

[Chamberlain 82]

Chamberlain D.C., King J.C., Slutz D.R., Todd S.J.P., Wade B.W.
"JANUS : An Interactive Document Formatter Based on Declarative Tags"
IBM Computer Science Research Report RJ3366(40402)
IBM Research Laboratory, June 1982

[Coulouris 76]

Coulouris G.F.
"The Design and Implementation of an Interactive Document Editor"
Software - Practice and Experience 6(2) : 271 - 279
June 1976

[EPFL 81 a]

Laboratoire de Micro-informatique (E.P.F.L.)
"Smaky"
Documentation de l'Ecole Polytechnique Fédérale de Lausanne
Switzerland, August 1981

[EPFL 81 b]

Laboratoire de Microinformatique (E.P.F.L.)
"Sanders.Tx"
Documentation de l'Ecole Polytechnique Fédérale de Lausanne
Switerzland, October 17, 1981

[Goldberg et Kay 76]

Goldberg A., Kay A.
"Smalltalk - 72 Instruction Manual"
Report No.SSL-76-6
Xerox Alto Research Center, March 1976

[Goldfarb 81 a]

Goldfarb C.F.
"Use of an Integrated Text Processing System in a Commercial Textbook Production"
Abstract of the Presented Papers, International Conference on Research and Trends in Document Preparation Systems, Lausanne : 121 - 122
Switerzland, February 1981

[Goldfarb 81 b]

Goldfarb C.F.
"A Generalized Approach to Document Markup"
Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices 16(6) : 68 - 73
August 1981

[Good 81]

Good M.
"An Ease of Use Evaluation of An Integrated Editor and Formatter"
Technical Report MIT/LCS/TR-266
Massachusetts Institute of Technology Laboratory for Computer Science, August 1981

[Hammer 81 a]

Hammer M., Ilson R., Anderson T., Gilbert E.J., Good M., Niamir B., Rosenstein L., Schoichet S.
"An Integrated Document Proccesing System"
Office Automation Group Memo OAM 028
Massachusetts Institute of Technology Laboratory for Computer Science, March 1981

[Hammer 81 b]

Hammer M., Ilson R., Anderson T., Gilbert E.J., Good M., Niamir B., Rosenstein L., Schoichet S.
"The Implementation of Etude, an Integrated and Interactive Document Production System"
Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices 16(6) : 137 - 141
June 1981

[Hersch 82]

Hersch R.D.
"Formatted Document Description"
Documentation de l'Ecole Polytechnique Fèderale de Lausanne
Switerzland, June 1982

[Hersch 82]

Hersch R.D.
"Generating and Printing of High Quality"
"Document in a Distributed Office Automation Network Environment"
Documentation de l'Ecole Polytechnique Fèderale de Lausanne
Switerzland, January 1981

[IBM 80a]

IBM
"Facility - Introduction of the Generalized Markup Language, using the Starter Set"
IBM Corporation, Order number SH20 - 9186 - 0
1980

[IBM 80b]

IBM
"Document Composition Facility Generalized Markup Language : Starter set Reference"
IBM Corporation, Order number SH20 - 9187 - 0
1980

[Ilson 80]

Ilson R.
"An Integrated Approach to Formatted Document Production"
Technical report MIT/LCS/TR - 253
Massachusetts Institute of Technology Laboratory for Computer Science, August 1980

[Ingalls 78]

Ingalls D.H.
"The Smalltalk-76 Programming System Design and Implementation"
Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages
January 1978

[Kernyhan, Lesk et Ossanna 78]

Kernyhan B.W., Lesk M.E., Ossanna J.F.
"UNIX Time-Sharing System : Document Preparation"
The Bell System Technical Journal 57(6) : 2115 - 2135
July-August 1978

[Knuth 79]

Knuth D.E.
"Tex and Metafont"
Digital Press and the American Mathematical Society
1979

[Lampson 78]

Lampson B.W.
"Bravo Manual"
In Lampson B.W. and Taft E.A., Alto Users Handbook. Computer science Laboratory
Alto Research Center, 1978

[LOGITECK 82]

LOGITECK S.A.
"Raster printing Systems for Graphic Arts Applications"
No (11) : 111 - 123
June 14, 1982

[MacKay 77]

MacKay P.A.
"Setting Arabic with a Computer"
Scholarly Publisching 8(2) : 142 - 150
January 1977

[NEW ENTERPRISE DIVISION 80]

NEW ENTERPRISE DIVISION
"Canon Laser Beam Printer LBP - 10"
Mai 1980

[Nicoud 82]

Nicoud J.D., Nievergelt J., Coray G., Show A.C.
"Document Preparation Systems"
North Hollands Publishing Company 1982

[Ossanna 74]

Ossanna J.F.
"NROFF - User's Manual - second edition"
International Document
Bell laboratories, September 1974

[Reid 80a]

Reid B.K.
"A High-Level Approach to Computer Document Foematting"
Conference Record of the Seventh Annual ACM Symposium on Principles of Programmings Languages
January 1980

[Reid 80b]

Reid B.K.
"SCIBE : A Document Specification Language and its Compiler"
Ph. D. thesis, Carnegie-Mellon University, Computer Science Departement
Pittsburgh, October 1980

[Reid 81]

Reid B.K.
"The Sribe Document Specification Language and its Compiler"
Abstracts of the Presented Papers, International Conference on Research and Trends in Document Preparation Systems
Switzerland, February 1981

[Reid et Walker 80]

Reid B.K., Walker J.H.
"SCRIBE Introductory User's Manual"
Third Edition, Preliminary Draft.
UniLogic, Ltd, Pittsburgh 1980

[Saltzer 65]

Saltzer J.
"Manuscript Typing and Editing : TYSET, RUNOFF"
In Crisman P.A., editor, The Compatible Time-sharing System : A Programmer's Guide, Second Edition, section AH.9.01., MIT Press 1965

[Seybold 81]

Seybold J.
"Xerox's 'Star'"
The Seybold Report 10(16)
April 27, 1981

[Shock 79]

Shock J.F.
"An Overview of the Programming language Smalltalk-72"
SIGPLAN Notices 14(9) : 64 - 73
September 1979

[Simpson 77]

Simpson R.M.
"Macros for Course Notes and Technical Report"
January 7, 1977

[Smith 82]

Smith D.C., Irby C., Kimball R., Verplank B.
"Designing the Star User Interface"
Byte 7(4) : 242 - 282
April 1982

[Sommers 81]

Sommers R.
"A Real Time Protocol for a Sublocal Network"
Documentation de l'Ecole Polytechnique Fèdèrale de Lausanne
Switerzland, Mai 13, 1981

[Tesler 72]

Tesler L.
"PUB : The Document Compiler"
Operating Note 70, Standford Artificiel Intelligence Project
September 1972

[Thacker 79]

Thacker C.P., MacCreight E.M., Lampson B.W., Sproull R.F., Boggs D.R.
"Alto : A Personnel Computer"
Technical Report CSL - 79 - 11, Xerox Palo Alto Research Center.
August 1979
1979

[Thompson et Ritchie 75]

Thompson K., Ritchie D.M.
"UNIX Programmer's Manual"
Bell Telephone Laboratories, Inc., 1975

[Van Lamsweerde 81]

Van Lamsweerde A.
Notes de cours : "Analyse Organique"
1981

[Van Wyk 80]

Van Wyk C.J.
"A Language for Typesetting Graphics"
Ph. D. , thesis, Standford University, California, June 1980

[VISUAL TECHNOLOGY INCORPORATED 80]

VISUAL TECHNOLOGY INCORPORATED
"Video Display Terminal - Reference Manual"
Tewksbury, July 1980